

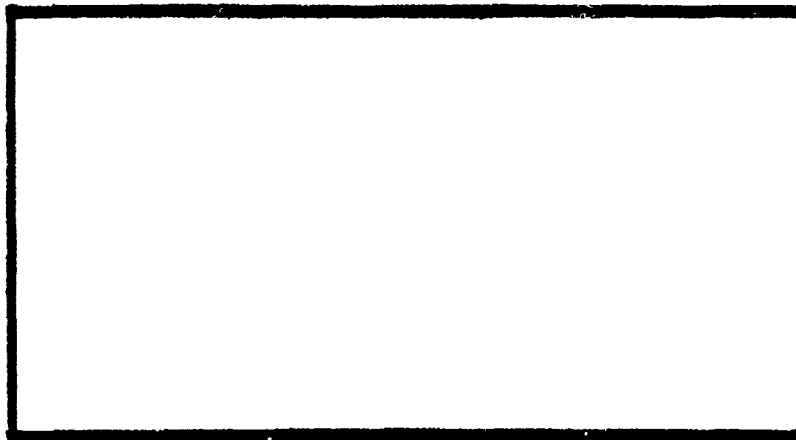
AD-A243 797



1



DTIC  
ELECTE  
DEC 30 1991  
S D



This document has been approved  
for public release and sale; its  
distribution is unlimited.

DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY  
**AIR FORCE INSTITUTE OF TECHNOLOGY**

Wright-Patterson Air Force Base, Ohio

AFIT/GSO/ENS/91D-15

DTIC  
ELECTE  
DEC 30 1991  
S D D

BINARY PROGRAMMING MODELS OF SPATIAL  
PATTERN RECOGNITION: APPLICATIONS IN  
REMOTE SENSING IMAGE ANALYSIS

THESIS

Thomas Gerald Reed  
Captain, USAF

AFIT/GSO/ENS/91D-15

This document has been approved  
for public release and sale; its  
distribution is unlimited.

Approved for public release; distribution unlimited

91-19031



91 12 24 055

AFIT/GSO/ENS/91D-15

BINARY PROGRAMMING MODELS OF SPATIAL PATTERN  
RECOGNITION: APPLICATIONS IN REMOTE SENSING  
IMAGE ANALYSIS

THESIS

Presented to the Faculty of the School of Engineering  
of the Air Force Institute of Technology  
Air University

In Partial Fulfillment of the  
Requirements for the Degree of  
Master of Science in Space Operations

Thomas Gerald Reed, B.S.  
Captain, USAF

December, 1991



Accession For	
NTIS CR&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification .....	
By .....	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

Approved for public release; distribution unlimited

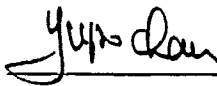

## THESIS APPROVAL

*STUDENT* : Captain Thomas G. Reed

*CLASS* : GSO-91D

*THESIS TITLE* : BINARY PROGRAMMING MODELS OF SPATIAL PATTERN  
RECOGNITION: APPLICATIONS IN REMOTE SENSING IMAGE ANALYSIS

*DEFENSE DATE* : 22 November 1991

COMMITTEE	NAME/DEPARTMENT	SIGNATURE
Advisor	Dr. Yupo Chan/ENS	 _____
Reader	Major Thomas S. Kelso/ENS	 _____



## *Acknowledgments*

First and foremost, I wish to thank my parents, Dr. and Mrs. James L. Reed, Jr., for giving me the opportunity and support that have allowed me to enjoy the successful career and life I now have. I am very grateful to my wife, Suzanne, for her continuing support of my work and school efforts while maintaining the kind of family lifestyle I've grown accustomed to (namely, bringing home that second pay check). I wish to thank my future child, "pre-partum", who helped remind me to keep everything in perspective.

As for my actual thesis effort, I would not have been able to accomplish as much as I did were it not for the guidance and instruction I received from my thesis advisor, Dr. Yupo Chan, and thesis reader and class advisor, Major Thomas S. Kelso. It was their extensive knowledge of the underlying principles behind my research that allowed me to quickly step into a world that was formerly alien to me and accomplish work deserving of the name "thesis". Thanks, too, to Dan Burke, whose work and instruction got me started on my thesis.

I believe a special thanks is due Major Roth whose  $\text{\LaTeX}$  templates enabled all of us to concentrate on the substance of our theses rather than on their appearance and layout and still be able to produce professional-looking documents with little effort. I wish to thank all the past and present staff and faculty who managed to procur all the state-of-the-art computer hardware, software, mainframe systems, and peripherals here at AFIT that rival those at any other educational institution. Finally, I would like to thank the other faculty members, secretaries, squadron staff, and computer room folks who made my stay here easier, and the cleaning, civil engineering, and AAFES folks who made it bearable.

Thomas Gerald Reed

## *Table of Contents*

	Page
Acknowledgments . . . . .	ii
Table of Contents . . . . .	iii
List of Figures . . . . .	x
List of Tables . . . . .	xiii
Abstract . . . . .	xiv
 I. Introduction . . . . .	 1
1.1 Image Classification . . . . .	1
1.2 Problem Statement . . . . .	2
1.3 Background . . . . .	2
1.3.1 Image Segmentation. . . . .	2
1.3.2 Clustering. . . . .	3
1.4 Sub-Objectives . . . . .	5
 II. Literature Review . . . . .	 6
2.1 Introduction . . . . .	6
2.2 Visual . . . . .	6
2.3 Mathematical Morphology . . . . .	6
2.4 Split and Merge . . . . .	7
2.5 Hierarchical Stepwise (Beaulieu) Optimization Segmentation Algorithm . . . . .	8
2.6 Geometric Primitives . . . . .	9
2.6.1 Multi-Shape Detection. . . . .	9

	Page
2.6.2 Line Segment Extraction and Re-Combination. . .	9
2.6.3 Planimetric Feature Extraction. . . . .	10
2.6.4 Line Segment Extraction From Statistical Texture Analysis. . . . .	11
2.6.5 Edge Following as Graph Searching and Hough Trans- forms. . . . .	12
2.7 Image Processing . . . . .	12
2.8 Summary . . . . .	13
III. Binary Programming Formulations of the Subregion Allocation Problem	14
3.1 Districting . . . . .	14
3.2 Single Subregion Allocation (SSA) . . . . .	14
3.3 Single Subregion Allocation Example . . . . .	18
3.4 Multiple Subregion Allocation (MSA) . . . . .	19
3.5 Multiple Subregion Allocation Example . . . . .	22
3.6 A Final Single Subregion Allocation Example . . . . .	24
3.7 Applicability and Complexity of the B&W Subregion Allo- cation Models . . . . .	25
3.8 The Modified Subregion Allocation Models . . . . .	28
3.9 Modified Single Subregion Allocation (MSSA) . . . . .	28
3.10 Modified Single Subregion Allocation Example . . . . .	31
3.11 Modified Multiple Subregion Allocation (MMSA) . . . . .	32
3.12 Modified Multiple Subregion Allocation Example . . . . .	33
3.13 A Final Modified Single Subregion Allocation Example . .	40
3.14 Applicability and Complexity of the Modified Subregion Al- location Models . . . . .	44

	Page
IV. Network Formulations of the Subregion Allocation Problem . . . . .	45
4.1 The Area-Wall Network Formulation of the Single Subregion Allocation Problem . . . . .	45
4.1.1 Area-Wall Network Example 1 (AREAWALL1). . . . .	48
4.1.2 Area-Wall Network Example 2 (AREAWALL2). . . . .	49
4.2 The Area-Wall Network Formulation of the Multiple Subre- gion Allocation Problem . . . . .	54
4.3 Conclusions of the Area-Wall Network . . . . .	58
4.4 Version 1 of The Cell-Border Network Formulation of the Single Subregion Allocation Problem . . . . .	60
4.5 Version 2 of The Cell-Border Network Formulation of the Single Subregion Allocation Problem . . . . .	70
4.6 Determining the Most Compact Subregion . . . . .	78
4.7 Conclusions . . . . .	83
V. Pixel Subregion Allocation: The Use of Subregion Allocation Concepts Towards Image Analysis . . . . .	84
5.1 Purpose . . . . .	84
5.2 The GAMS Single Subregion Allocation Model . . . . .	85
5.3 The Programs and Files . . . . .	87
5.4 The Example Runs . . . . .	88
5.5 Subregion Allocation of Real Satellite Imagery . . . . .	88
5.6 Conclusions . . . . .	91
VI. Conclusions and Recommendations . . . . .	94
6.1 Subregion Allocation Through Binary Programming . . . . .	94
6.2 Subregion Allocation Through Networks . . . . .	95
6.3 The Multiple Objectives of Subregion Allocation . . . . .	99
6.4 The Spatial Analysis of Imagery . . . . .	99
6.5 Final Recommendations and Conclusions . . . . .	100

	Page
Appendix A. Instructions for Running the Computer Programs . . . . .	102
A.1 BP Programs . . . . .	102
A.1.1 Running a GAMS Program. . . . .	102
A.1.2 The Burke Implementation of the B&W Models. . . . .	102
A.1.3 The Imaging Programs. . . . .	103
A.2 Network Programs . . . . .	104
A.2.1 Running a SAS Program. . . . .	104
A.2.2 Running the Area-Wall Network Program. . . . .	104
A.2.3 Running the Cell-Border Network Program. . . . .	105
Appendix B. Files for the B&W and Modified Subregion Allocation Problems . . . . .	106
B.1 Partial Output File for Problem SSA1 . . . . .	106
B.2 Input File for Problem MSA1 . . . . .	110
B.3 Partial Output File for Problem MSSA1 . . . . .	113
B.4 Input File for Problem MMSA2 . . . . .	115
B.5 Partial Output File for Problem MMSA4 . . . . .	117
B.6 Partial Output File for Problem MSSA3 . . . . .	121
Appendix C. Programs and Files for The Network Formulations of the Subregion Allocation Problems . . . . .	125
C.1 The "SUBNETS" Pascal Program . . . . .	125
C.2 The "AREAWALL" Pascal Program . . . . .	134
C.3 The "CELLBORDER" Pascal Program . . . . .	145
C.4 Input File for Problem AREAWALL1 . . . . .	169
C.5 Partial .log and .lis Output Files for Problem AREAWALL1 . . . . .	173
C.6 Partial .log and .lis Output Files for Problem AREAWALL2, Run 1 . . . . .	175

	Page
C.7 Partial .log and .lis Output Files for Problem AREAWALL2, Run 2 . . . . .	178
C.8 Partial .log and .lis Output Files for Problem AREAWALL2, Run 3 . . . . .	180
C.9 Partial .log and .lis Output Files for Problem AREAWALL2, Run 4 . . . . .	182
C.10 Partial .log and .lis Output Files for Problem AREAWALL2, Run 5 . . . . .	184
C.11 Partial .log and .lis Output Files for Problem AREAWALL2, Run 8 . . . . .	185
C.12 Partial .log and .lis Output Files for Problem AREAWALL3, Run 1 . . . . .	187
C.13 Partial .log and .lis Output Files for Problem AREAWALL3, Run 2 . . . . .	190
C.14 Partial .log and .lis Output Files for Problem AREAWALL3, Run 3 . . . . .	192
C.15 Partial .log and .lis Output Files for Problem CELLBOR- DER1, Run 1 . . . . .	194
C.16 Input File for Problem CELLBORDER1, Run 2 . . . . .	196
C.17 Partial .log and .lis Output Files for Problem CELLBOR- DER1, Run 2 . . . . .	211
C.18 Partial .log and .lis Output Files for Problem CELLBOR- DER2A, Run 1 . . . . .	213
C.19 Partial .log and .lis Output Files for Problem CELLBOR- DER2B, Run 1 . . . . .	216
C.20 Partial .log and .lis Output Files for Problem CELLBOR- DER2B, Run 2 . . . . .	217
C.21 Partial .log and .lis Output Files for Problem CELLBOR- DER2B, Run 3 . . . . .	218
C.22 Partial .log and .lis Output Files for Problem CELLBOR- DER2B, Run 5 . . . . .	220

	Page
C.23 Input File for Problem CELLBORDER2B, Run 6 . . . . .	222
C.24 Partial .log and .lis Output Files for Problem CELLBOR- DER2B, Run 6 . . . . .	242
C.25 Partial Input File for Problem CELLBORDER2B, Run 7 .	245
C.26 Partial .log Output File for Problem CELLBORLER2B, Run 7 . . . . .	250
C.27 Partial Input File for Problem CELLBORDER2B, Run 8 .	251
C.28 Partial .log Output File for Problem CELLBORDER2B, Run 8 . . . . .	257
C.29 Partial Input File for Problem CELLBORDER2B, Run 9 .	258
C.30 Partial .log Output File for Problem CELLBORDER2B, Run 9 . . . . .	264
C.31 Partial Input File for Problem CELLBORDER2B, Run 10	265
C.32 Partial .log and .lis Output Files for Problem CELLBOR- DER2B, Run 10 . . . . .	270
 Appendix D. Pixel Subregion Allocation Computer Programs, Input and Output Files . . . . .	 271
D.1 The "IMAGING" System Program . . . . .	271
D.2 The "SOLUTION" Pascal Program . . . . .	272
D.3 The Complete Screen Output of the Pseudo-Image Low Com- pactness Example Run . . . . .	276
D.4 The GAMS Input File for the Low Compactness Example Run . . . . .	280
D.5 The Partial Screen Output of the Pseudo-Image High Com- pactness Example Run . . . . .	281
D.6 The Partial Screen Output of the Real Image High Compact- ness Example Run . . . . .	282
D.7 The Partial Output File of the Real Image Example Run .	284

	Page
Appendix E.     The Burke Implementation of the B&W Models . . . . .	285
E.1   A Few Words Concerning the Work of Burke . . . . .	285
E.2   The "MSA" System Program . . . . .	286
Bibliography . . . . .	287
Vita . . . . .	289



## *List of Figures*

Figure	Page
1. Neighbors of a Matrix Cell. . . . .	16
2. SSA1:16-Cell Single Subregion Allocation Example Input. . . . .	18
3. SSA1:16-Cell Single Subregion Allocation Example Solution. . . . .	19
4. MSA1:36-Cell Multiple Subregion Allocation Example Input. . . . .	23
5. MSA1:36-Cell Multiple Subregion Allocation Example Solution. . . . .	23
6. SSA2:36-Cell Single Subregion Allocation Example Input. . . . .	24
7. SSA2:36-Cell Single Subregion Allocation Example Solution. . . . .	25
8. MSSA1:16-Cell Modified Single Subregion Allocation Example Input. . . . .	31
9. MSSA1:16-Cell Modified Single Subregion Allocation Example Solution. . . . .	32
10. MMSA1:36-Cell Modified Multiple Subregion Allocation Example Input. . . . .	33
11. MMSA1:36-Cell Modified Multiple Subregion Allocation Example Solution 1. . . . .	34
12. MMSA2:36-Cell Modified Multiple Subregion Allocation Example Input. . . . .	35
13. MMSA2:36-Cell Modified Multiple Subregion Allocation Example Solution 2. . . . .	36
14. MMSA3:36-Cell Modified Multiple Subregion Allocation Example Solution 3. . . . .	37
15. MMSA4:36-Cell Modified Multiple Subregion Allocation Example Solution 4. . . . .	40
16. MSSA2:36-Cell Single Subregion Allocation Example Input. . . . .	40
17. MSSA2:36-Cell Single Subregion Allocation Example Solution 1. . . . .	41
18. MSSA3:36-Cell Single Subregion Allocation Example Solution 2. . . . .	42
19. Area-Wall Network Structure for a 3-cell Min-Cost SSA Problem. . . . .	47
20. AREAWALL1:16-Cell Single Subregion Allocation Example Input. . . . .	48
21. AREAWALL1:16-Cell Single Subregion Allocation Example Solution. . . . .	49

Figure	Page
22. AREAWALL2:16-Cell Single Subregion Allocation Example Input. .	50
23. AREAWALL2:16-Cell Single Subregion Allocation Example Solution.	52
24. Area-Wall Network Structure for a 5-Cell and 3-Cell Min-Cost MSA Problem. . . . .	55
25. AREAWALL3:16-Cell Multiple Subregion Allocation Example Input.	56
26. AREAWALL3:Allocation Results from Run Number 1. . . . .	57
27. AREAWALL3: 16-Cell Multiple Subregion Allocation Example Solution. . . . .	58
28. Break-Down of a Data Matrix into its Cell-Border Nodes. . . . .	62
29. Lower-Right-Hand Corner of the Example Cell-Border Network. . . .	63
30. CELLBORDER1: 16-cell Single Subregion Allocation Example Input.	64
31. CELLBORDER1: 16-Cell Single Subregion Allocation Example Solution. . . . .	65
32. CELLBORDER2: 16-Cell Single Subregion Allocation Example Input.	66
33. CELLBORDER2A: 3-Row-by-2-Column Single Subregion Allocation Solution. . . . .	66
34. CELLBORDER2B: Allocation Results From Run Number 1. . . . .	67
35. CELLBORDER2B: Allocation Results From Run Number 3. . . . .	68
36. CELLBORDER2B: Allocation Results From Run Numbers 4 and 5.	70
37. CELLBORDER2B: Allocation Results From Run Number 6. . . . .	72
38. Allocation Cost Assessment of the Cell-Border Network Version 2. .	74
39. Non-Allocation Cost Assessment of the Cell-Border Network Version 2.	75
40. The Cell-Border Single Optimum vs. the B&W Multiple Optimum Solutions. . . . .	76
41. CELLBORDER2B: Allocation Results From Run Number 7. . . . .	77
42. CELLBORDER2B: Allocation Results From Run Number 8. . . . .	77
43. Final Non-Allocation Cost Assessment Example. . . . .	79
44. CELLBORDER2B: Allocation Results From Run Number 9. . . . .	79

Figure	Page
45. Size-7 Subregions Possible When Shape Constraints Are Not Specified.	81
46. CELLBORDER2B: Rounded-off Allocation Results From Run Number 10. . . . .	82
47. The Pseudo-Image Input Data and (Frame-less) Working Data. . . .	88
48. Pseudo-Image Solutions for Low and High Compactness Options. . .	89
49. The Real Image Input. . . . .	89
50. The Real Image (Frame-less) Working Data. . . . .	90
51. The Real Image Solution for the High Compactness/Contiguity Option.	92

## *List of Tables*

Table	Page
1. A Comparison of the B&W and Modified Subregion Allocation Models.	44
2. Flow Results of Problem AREAWALL2, Run Number 1. . . . .	51
3. Flow Results of Problem AREAWALL2, Run Number 2. . . . .	53
4. Flow Results of Problem AREAWALL2, Run Number 3. . . . .	53
5. Flow Results of Problem AREAWALL2, Run Number 5. . . . .	53
6. Flow Results of Problem AREAWALL2, Run Number 6. . . . .	54
7. Flow Results of Problem AREAWALL2, Run Number 8. . . . .	54

*Abstract*

The major purpose of this investigation was to implement subregion allocation objectives using a network model based on an existing subregion allocation binary programming model (Benabdallah and Wright (B&W), 1990), the ultimate goal being the application of subregion allocation concepts towards the spatial analysis of satellite imagery.

The multi-objective aspects of subregion allocation can be accomplished via a network formulation, a formulation vastly simpler in complexity than the binary programming models previously used. Without a network programming package that could maintain integral flows, however, deriving the solution was a tiresome task for the user. Nonetheless, several new concepts and advantages to using networks were discussed and demonstrated to the degree possible.

Spatial analysis could benefit from applying a modified version of the B&W model to the pixels of satellite imagery in a way which takes advantage of the inherent contiguity of natural terrain subregions. Trending and forecasting of subregion changes, tasks that rely on acquiring data in a consistent manner image after image, could benefit due to the fact that major spatial characteristics of subregions could be extracted, and minor spatial changes could be removed.

# BINARY PROGRAMMING MODELS OF SPATIAL PATTERN RECOGNITION: APPLICATIONS IN REMOTE SENSING IMAGE ANALYSIS

## *I. Introduction*

### *1.1 Image Classification*

Image classification and analysis are used for a variety of purposes, some of which are: classification of imaged terrain areas for geographic coding systems (6:7), analysis of terrain imagery for geological studies (10:85), (18:93), and recognition and location of objects (15:477), (13:208), (5:251). The primary goal of geographic coding systems is to categorize the pixels of an image "into land cover classes or themes" (6:28). One such classification scheme, the U.S. Geological Survey system, details over 30 different areas (such as residential, pasture, lakes, forested wetlands, bare exposed rocks, shrub and brush tundra, and glaciers) into which terrain may be classified (11:52-53). Geological studies employing satellite image analysis attempt to understand regional tectonics and aid in finding oil, gas, and mineral deposits (10:85). Finally, the area of analysis concentrating on object detection can be used in such areas as aerial navigation systems (15:477), robot and machine vision systems (13:208), industrial parts manufacturing (16:363-364), and automatic target identification and location (14:456).

Although much aerial and satellite image analysis is performed by human analysts (10:85), (18:93), "computer-based methods of analyzing digitized imagery" (14:456) are needed for a variety of reasons. One reason is to keep the overwhelming amount of imaged data from being mis-handled, lost, or wasted. "Future massive quantities of various types of collected imagery create an urgent need for automation of the image data handling and storage process" (14:456). Computer-based methods would also aid in the "storage and retrieval of image data for later analyses" (14:456) such as forecasting and trending subregion changes. These applications

have one requirement in common: the "extraction of shape information" (14:456), information which cannot be extracted without proper image segmentation and region and boundary determination (11:49), (9:15).

## *1.2 Problem Statement*

The major purpose of this investigation is to determine to what extent subregion allocation can be performed using a network model based on the Burke implementation of the Multiple Subregion Allocation (MSA) binary programming model (by Benabdallah and Wright (B&W), 1990). A secondary objective is to determine to what extent the MSA model itself can be used for the purposes of spatial analysis of satellite imagery. The ultimate goal is the development of a model which can segment actual satellite imagery into various geographic regions whose properties are described by the model user. Modifications to this general-purpose, multiobjective, programming model (1:34) (including constraint and objective modifications) will be developed and explained in Chapter III. The MSA, modified MSA, and network models will be used to segment into subregions cells of matrices containing cost data. Currently, the "MSA [model] ... cannot handle large size problems" (1:34).

## *1.3 Background*

*1.3.1 Image Segmentation* The segmentation of an image into homogeneous regions is one of the two fundamental processes in image analysis, the other being edge extraction (9:15). Segmentation of imagery into shapes and regions can be performed by using appropriate techniques based on spectral and spatial analysis. Spectral analysis is typically used to determine what is contained in an area of an image by basing the classification on the features' "inherent spectral reflectance and emittance properties" (6:28). Spatial analysis, on the other hand, is more often used to determine the physical attributes of some area of interest. Together, these two types of analyses can determine such characteristics as tone (shade variations), color, pattern, association, shape, shadow, size, and resolution (11:47). The successful determination of many of these characteristics depends on proper spatial analysis which, as has previously been referred to, depends on the detection of features such as area boundaries and edges (11:49).

Feature extraction, in general, depends on the ability to detect edges, or lineaments. "A lineament is a mappable, simple or composite linear feature of a surface

that is aligned in a rectilinear or slightly curvilinear relationship [and] that represents faults, fractures, joints, contacts, topographic linear ridges, valleys and tonal contrasts" (10:85). In general, lineaments are due to either "physiographic features ... or tonal changes" (10:85). To somewhat mirror the manual method of lineament detection, "based on color, tone, texture, size, pattern and the nature of the surroundings," an automatic lineament detection method should "take into consideration the noise, threshold, size and orientation of the lineaments of interest" (10:86). Geometric relationships, spatial attributes, and brightness can all be used to extract the desired line segments and sequentially remove those line segments not pertaining to the features of interest (19:2251). In this way, edges and corners and curved surfaces, and thus any feature which can be described by these lineaments, can be determined (19:2250).

*1.3.2 Clustering.* Clustering can be applied to geographic coding systems as a means to automatically perform the desired spatial analysis and districting (segmentation) of regions of interest (6:31). "The main purpose of most districting models is the design of a predetermined number of territories or districts with contiguous and compact shapes" (6:31). While methods do exist which will accomplish the districting for small problems, "for problems that involve clustering of all cells in a field ..., enumeration methods are infeasible" (6:35).

*1.3.2.1 Spatial Clustering.* The automatic clustering algorithm used by Béné and others "uses Euclidean distance as a measure of similarity [and] is based on the concept of spatial clustering" (2:5). There are six steps to this algorithm: (1) extract texture and edge features, (2) calculate local means and standard deviations for these featured images, (3) "determine the threshold and calculate the merging distance ' $d$ '" (2:5), (4) merge regions whose separation is less than the merging distance, (5) implement another algorithm to determine each region's nucleus, and (6) "classify each pixel into the set with the nearest nucleus" (2:5).

In this instance, the researchers found that they were limited in the number of classes that might be detected due to limitations in the "number and size of the input bands" (2:11), a spectral limitation. Further, because the segmentation was based more on a pixel-by-pixel pattern emphasis than on spatial relationship, accuracy was less than desired. A good feature was that processing time was relatively short overall. (2:7,9-10).



1.3.2.2 *Very Large Data Set Clustering for Astronomical Image Analysis.* Bruynooghe's research in one astronomical study explains the basic ideas behind the large data set clustering method he used. He states that "this new clustering strategy is based on the complementary utilization of an accelerated partitioning algorithm generating a large number of small clusters and then of an accelerated hierarchical clustering algorithm" (3:101). The Phase 1 partitioning algorithm "incorporates a fast nearest-neighbor algorithm" (3:103). Phase 2, hierarchical clustering, builds "a binary tree on the set of elementary clusters previously obtained" (3:103) and employs "an efficient branch-and-bound algorithm for the generation of a set of neighborhoods" (3:104). The method, capable of clustering over 10,000 pixels into hundreds of clusters in approximately 14 minutes (3:104), is 25 times faster than previous methods mostly due to "eliminating the necessity of calculating many pairwise distances" (3:104).

1.3.2.3 *Multiple Subregion Allocation.* As has been stated, the MSA models which will be used as a basis for this research are those which were developed by Benabdallah and Wright (1990) and further tailored and implemented by Burke. These individuals conducted research to increase the ability of a "multi-objective integer programming model" (6:31), a districting model which had been employed to segment a region based on population units (1:4). The first phase of the basic model "generates all feasible districts [based on] contiguity, compactness, and limited population deviation. The second phase finds the set of feasible districts that minimizes the maximum deviation from the mean district population and covers each population unit exactly once" (1:4). The initial model is restricted both in the shape the regions can assume (rectangular) and in the size of the initial problem (1:33-34).

The same constraints can be found in the updated nonlinear model and are defined as follows: "contiguity requires that each district be a connected subregion of land ...; compactness requires that the single subregion be consolidated rather than spread out" (1:17). Finally, "the main objective is to place these blocks into [a pre-determined number of] contiguous districts such that each district's population is within the district's population limits" (1:17).

Specifics on the B&W MSA model are contained in Chapter III. In the general MSA problem formulation:

The overall objective is a combination of the following objectives: minimize the cost of cell acquisition into a region, maximize the area of each region, and minimize

the perimeter of each region. (Recall that in this MSA model, the desired number and size of regions is already known.) (1:10)

This overall objective is subject to the following constraints:

- The model must keep track of which cells are assigned to which regions,
- Cells may belong to at most one region, and
- The variables used to meet the above constraint are binary variables (1:5-11).

As was stated in the previous section, the purpose of this research is to develop a general model capable of segmenting large numbers of satellite imagery pixels into multiple regions. Because the research objective is to develop a general segmentation model with applications towards geographic coding systems and not specifically a population districting model, the research model will contain different objectives and constraints than the MSA model and will focus on the binary formulations of the MSA model. The mathematical descriptions and development of the variables, objectives, and constraints of the research model will be discussed in Chapter III.

#### *1.4 Sub-Objectives*

The investigation will cover the following areas:

- Develop algorithms and computer programs to enact the B&W and modified MSA algorithms.
- Validate the MSA algorithms on the experimental data.
- Perform the segmenting and validating on progressively complex experimental data.
- Make final assessment as to whether, and in what specific applications, the modified MSA model shows improved segmentation ability over the B&W MSA model.
- Develop and implement network formulations of the basic subregion allocation model.
- Develop computer programs to enact the network models.
- Validate the network models on the experimental data.
- Apply subregion allocation concepts to actual satellite imagery.

## *II. Literature Review*

### *2.1 Introduction*

The many techniques available for performing image segmentation, can be grouped into three major categories by virtue of what the segmentation is based on: discontinuities (edge detection), similarities (region-based segmentation), and hybrid methods (combinations of the two) (15:483), (2:2), (9:15). The following sections present a discussion of the underlying concepts of major segmentation and feature extraction methods currently being used.

### *2.2 Visual*

Klasse used Landsat multispectral scanner system images to classify and measure oasis agricultural land in the Turpan Depression, China (11:8). Spectral analysis was used to determine the tones, colors, and, to a partial extent, the patterns and shades of the areas of interest. Spatial analysis was used mostly to determine area size (11:47-48).

The methods employed in Klasse's study to determine area and size involved first using a Bausch and Lomb Zoom Transfer Scope to enlarge the 1:1,000,000 scale photos to 1:250,000. Next, the researchers overlaid a scaled grid onto the photographic proof. "A visual estimate was made of the proportional amount of oasis agriculture in each cell of the grid" (11:15). At the scale used, the researchers were able to discriminate between areas approximately 10 acres in size (11:50,67). Edge detection was performed by visually observing tone contrast between areas (11:15,48-49).

### *2.3 Mathematical Morphology*

"Mathematical morphology, 'algebra of shape', is used in many image processing applications [and is] a tool for object recognition ... and feature extraction" (13:208). In general, mathematical morphology deals with the performance of the following operations: dilation (vector addition), erosion (vector subtraction), and combinations of the two (opening and closing) (13:211), (20:950). These operations are performed on two sets, those of the pixels of an image and those of "a structuring

element" (20:950), (13:211). These sets "are in Euclidean two-dimensional space for a binary image and in three-dimensional space for a grey-scale image" (20:950).

Lee and others, in their morphology study, combined the use of morphology and pyramid structuring of data to segment images for analysis. "A pyramid is a set consisting of an image, a registered sequence of its successive samplings, ... and the spatial relation between the different samplings" (13:209). Used as a method of implementing a multi-resolution (tree) representation of an image, the overall result is not only the ability to "represent large regions of an image [by single nodes]" (13:208), but also the capability to apply the two methods together "to image segmentation, object detection, [and] region (feature) description" (13:208). The image is successively simplified through different steps of the algorithm by taking advantage of "the morphological sampling theorem" (13:208-209), a theorem "analogous to the standard sampling theorem in signal processing [which states] that a signal can be completely reconstructed ... if the sampling rate ... is two times the highest frequency component of the signal" (13:211). The image is finally reduced to a pyramid representation of its pixels as "candidate fathers [and] candidate-sons" (13:209) of one another.

The experiments at image segmentation yielded positive results overall in the areas of segmentation error, rate of solution convergence, and computational complexity (13:216). Their techniques were applied to real and artificial images ("generated by controlling three parameters: image smoothness, image contrast, ... and the amount of noise" (13:215)). In general, as long as the signal-to-noise ratio was not too low (on the order of 6-8 dB), their techniques quickly and correctly segmented the images (13:218-219).

#### *2.4 Split and Merge*

"Split and merge is a region-based segmentation technique used mainly for object detection and image partitioning" (5:251). The general technique involves splitting the image into quadrants, each of which is split again and again until homogeneous regions are separated; the individual regions are then merged back into the original image taking into account various spectral and spatial factors (5:251,253). The split and merge method employs both types of region-based segmentation techniques: disjunctive (the splitting) and conjunctive (the spatial merging) (2:2). "[It] is basically a top-down approach with each level of split indicating the distinctive

features and the smoothness of the object boundaries" (5:253). "[It] is the most computationally efficient algorithm for region segmentation in terms of the number of searches in the spatial image grid due to the quadtree representation of the resultant regions" (5:252). The main drawback to the method is the "extensive computation time and large memory space" (5:252) required.

The researchers attempted to perform experiments in which parallel processors assumed much of the computational duties within the smaller partitions of the image, the desire being to reduce the overall computation time required to segment images and thus detect objects within the images. Although their initial efforts to distribute the computational workload among several processors proved successful (parallel processing dramatically reduced the time required to segment an image), their most promising approach has yet to be implemented (5:259).

### *2.5 Hierarchical Stepwise (Beaulieu) Optimization Segmentation Algorithm*

Using the Beaulieu algorithm, segmentation is accomplished by the use of the moments of the segments, the determination of segment boundaries and neighbors, and the stepwise optimization (2:4). There are four steps to this approach: (1) start with an initial segmentation of the image, (2) calculate the moments, boundaries, and neighbors of each segment, (3) calculate the cost of merging neighbors (cost criteria is dependent on the situation), and (4) perform the actual hierarchical stepwise merging (2:4). The merging step is an iterative step in which segment neighbors with minimum merging costs are merged until the desired number of segments is attained (2:4).

Believing that the requirement to know the number of segments contained in an image beforehand detracts from the algorithm's usefulness, the researchers modified this algorithm so that the number of segments needed to properly partition the image would be determined automatically. This modification was implemented using "the Student-t test on the difference of the means and a correction factor" (2:4). This modified Beaulieu algorithm has many positive points: no preparatory time before execution is required; it produces relatively good results in the classification of agricultural field data (though sometimes over-segmenting); its accuracy is fairly consistent (2:10).

## 2.6 Geometric Primitives

A primitive is simply a basic geometric "convex planar shape ... such as a circle, triangle, square, hexagon [or] polygon" (16:364). A segmentation or object detection method using primitives will "find the contour of the object [of interest], derive its curvature, locate the concavities ..., determine the type of primitive located in that region, and repeat the process until all concavities have been scanned" (16:365). By the time the image has been completely analyzed, the algorithm should have detected all regions or objects resembling the shape of the primitives under consideration (16:365).

*2.6.1 Multi-Shape Detection.* Differently shaped objects have been detected and identified by comparing peaks of the curvature functions to known peaks representing the different shapes (16:369-370). "The most critical operations in the process ... are the edge detection and the curvature computation of the contour of the object" (16:366). The researchers used a "Linear-Median Hybrid filter ... which combines nonrecursive linear filters and a median filter for the purpose of edge detection; ... the response from [the] edges [was] similar to that of standard edge detectors" (16:367). A curvature function was used to determine the curvature of the contours because, being a function of arc length, "the curvature uniquely describes [a continuously differentiable boundary curve]" (16:368). Further, smoothing was used to aid in focusing on determining "points of high curvature" (16:368) since these points contain most of the visual information about the curvature (16:368).

The researchers in the multi-shape detection experiments had high success in determining the location of and identifying several objects. Advantages of this method include its high processing speed and "its rotation and translation invariance" (16:375). Further, this method "can be applied to any set of regularly shaped objects by selecting the geometric primitives that best describe the objects of the set" (16:375).

*2.6.2 Line Segment Extraction and Re-Combination.* A study sponsored by Hughes Aircraft Co. incorporating the use of geometric primitives involves the use only of the line segment primitive to find specific targets: airplanes. Through three different processing levels, the system "extracts line segments from raw image data, ... combines the line segments into geometric feature primitives, and ... checks for positional coincidences of features to perform the airplane detections" (14:456).

The first level uses a new (Hughes algorithm) technique to "extract line segments from along the boundaries of major intensity changes" (14:458). The second level performs the grouping of the segments based on angle and position and "[represents the segments] symbolically by records containing endpoint coordinates, line segment angle, and line segment length" (14:458). The end results of this second stage is the generation of quadrilaterals (14:458). The third level compares the quadrilaterals generated by the second level to determine if they comprise the familiar shapes of airplanes (14:459). In order to have a positive identification of an airplane, the shapes representing the candidate airplane must meet three tests: reasonable feature positional coincidence, "plausible geometry" (14:460) of the feature groups, and presence of other minor features characteristic of airplanes (14:460).

The advantages to this system are similar to those of the previous method: no prior scene knowledge is required, object orientation is of little consequence, processing speed is high, and "low contrast conditions and limited occlusion can be tolerated" (14:456). Specifically, this method has additional advantages. Due to the three-level design, each level can perform its function very well, not having to concern itself with the overall purpose of the system (14:460). Results of the experiments have been positive: the Hughes line detection algorithm not only detected all line segments correctly under varying contrast intensities but also performed orders of magnitude faster than two other algorithms used for comparison (14:461-466). The researchers state that more research is warranted in areas such as using primitives in addition to the line segment in Level 1 and in detecting objects composed of more complicated shapes in Level 3 (14:466).

*2.6.3 Planimetric Feature Extraction.* The term *planimetric feature* refers to "geographic features [such as] roads, shorelines, and streams" (19:2250). These can be extracted from satellite imagery using shape primitives as discussed earlier. An often-performed planimetric feature extraction is that of road detection (12:2246).

Although road detection from satellite imagery is generally considered a spectral analysis problem, the problem can be approached as a linear feature detection problem whenever road width is less than the spatial resolution of the particular satellite sensor (12:2246). Because of this fact, road detection algorithms are analogous to lineament detection algorithms.

The researchers in the road detection study first experimented with different enhancement methods because contrast similarities between roads and backgrounds

prevent linear detection without enhancement (12:2246). They determined that using a false color composite of the bands with the highest clarity provided the best means of enhancement (12:2246).

The next step was to employ templates which respond to linear features. First and second-order neighbors of pixels were used to detect and extract road pixels from the imagery. Threshold values for use in determining if a pixel was a road pixel or not were established by using "the modes of the image grey value histogram" (12:2247). Once detected, the road pixels were linked. The researchers say that any of a number of linking procedures in use today are suitable (12:2247).

The results of this study indicate that most of the work involved in road detection lies in the pre-processing techniques of the imagery to enhance road features, techniques which cannot be determined with certainty beforehand. The advantages of this method are "its ability to detect roads irrespective of ... background (dark or light), [its] ease of implementation, and [its ability to be used as a] guideline for threshold setting" (12:2247).

*2.6.4 Line Segment Extraction From Statistical Texture Analysis.* This line segment extraction study conducted by Sudibjo and others focused on the use of synthetic aperture radar images for linear feature extraction (17:2242). Due to the fact that the brightness of features from this imagery highly depends on the angle at which the image was taken, the researchers hoped to implement an "auto-correlation function ... to measure the spatial variability (texture) in an image [and] optimize this texture analysis process to segment for linear feature detection" (17:2242).

The first steps involved removing as much noise as possible (speckle reduction) from the images through the employment of spatial filters. Five different filters were compared: "mean, median, mode, nearest values and minimum variance" (17:2243). Once filtered, the pixels were grouped based on their geometry as well as on their "spatial grey tone co-occurrence probabilities, grey tone run lengths, and textural edgeness" (17:2242). Four measures to extract textural information were used: contrast, uniformity, correlation, and entropy, each measure having its own unique mathematical formulation. The end result of the experimentation using all the various combinations of filters was that the median filter was best insofar as edge detection and extraction are concerned (17:2244).



*2.6.5 Edge Following as Graph Searching and Hough Transforms.* Edge Following as Graph Searching (EFGS) "can be used to extract edge magnitude and directions to produce an edge image; ... the Hough Transform identifies straight lines which represent edges" (18:93). The procedure steps are as follows: (1) smoothing and noise removal using a  $3 \times 3$  pixel-sized median filter, (2) implementation of EFGS, (3) use of the Hough transform, and (4) production of the lineament map (18:94).

## *2.7 Image Processing*

Many methods of filtering, combining, smoothing, and enhancing images prior to performing any analyses on them exist. This section will cover information on the SPOT satellite and pre-processing of the SPOT imagery used in the final experimental runs of Chapter V. All information on the satellite and the image processing was provided courtesy of Kelso (8).

The SPOT satellite is an earth-observing, French satellite that was launched in 1986 into an orbit with an altitude of approximately 517 miles. The SPOT image data used in Chapter V is the result of merging and processing the satellite's 20-meter resolution multispectral data and its 10-meter panchromatic data of four scenes of the Washington, D.C. area acquired between the dates of March 19 and March 24, 1987. The preparation of the data was performed by the EROS Data Center, Sioux Falls, South Dakota. To quote Kelso (8)

To accomplish the merge, an image-to-image registration was first performed between each scene set. Approximately 60 control points were selected for each set using an autocorrelation technique. The multispectral data were geometrically rectified and spatially resampled to 10-meter pixel ground resolution during this procedure to match the panchromatic data using a second-order polynomial and cubic convolution resampler. Once the panchromatic and multispectral data sets were geometrically identical, the merge was performed using a hue-intensity-saturation (HIS) method. The data merge was effected by transforming a contrast-enhanced spectral image into HIS color space.

The panchromatic data were stretched to match the distribution histogram of the spectral intensity channel, then substituted for the intensity channel to create a hybrid HIS data set. The hybrid HIS data was then transformed back into the red, green, and blue components. The four merged scenes were mosaicked together; each scene was registered

to 1:24,000 scale maps by selecting approximately 10 control points per image. These control points were combined with tie points selected by Large Area Mosaicking Software (LAMS) along adjacent scene boundaries and a transformation grid was created. The grid was used to geometrically correct each image in relation to the Universal Transverse Mercator (UTM) projection and to each other. Once each scene was in proper geometric configuration, the radiometric matching between scenes was accomplished using LAMS. Final processing of the data included a  $3 \times 3$  edge enhancement and multipoint linear stretch to increase contrast. The digital data were then output on a recorder in the form of a color transparency [which was subsequently] scanned on a graphic arts scanner [to generate] digital color separates.

Information on the data scenes can be obtained from Robert Lees, SPOT Image Corporation, Reston, VA (703-620-2200). Information on the color separates can be obtained from the Geological Survey offices in Reston, VA, and Denver, CO, and from various commercial vendors that sell USGS maps and publications. Finally, information on the imagery processing techniques referred to above can be obtained from the U.S. Geological Survey, EROS Data Center, Sioux Falls, South Dakota, 57198. The SPOT data presented here are copyrighted and were obtained under license from the SPOT Image Corporation, Reston, VA.

## *2.8 Summary*

The segmentation and feature extraction methods discussed above, as well as the clustering methods discussed in Chapter I, illustrate the wide range of techniques available to researchers. Many of the current studies in these areas focus on combining different image analysis techniques so as to improve the overall efficiency, effectiveness, and speed of the process of image analysis in general. Specifically, the literature review has revealed the combination of techniques in three areas: (1) implementing or developing appropriate image pre-processing techniques (filtering and smoothing) prior to performing any actual segmentations or feature extractions, (2) using parallel processing, tree structures, and hierarchical techniques to improve a method's performance by reducing and sharing the overall computation load, and (3) combining different segmentation or feature extraction techniques together to develop algorithms that are more robust than those that would be available through the use of the "baseline" or single techniques alone. Chapter III of this thesis will discuss in detail the clustering approach used as the basis of this research.

### *III. Binary Programming Formulations of the Subregion Allocation Problem*

The MSA model, including objective function and constraints, was briefly described in Chapter I. The following discussion will explain the mathematical development of the objectives and constraints as well as discuss such issues as model applicability and complexity. Solutions for the example runs were derived from the Zero/One Optimization Method (ZOOM) solver, a mixed-integer solver, via the use of the General Algebraic Modeling System (GAMS) within a VAX/VMS operating environment (4). Specifics on how to run a GAMS program can be found in Appendix A. Partial input and output files for the GAMS runs are on disk, with a select number of them as noted located in Appendix B. The B&W models were implemented using procedures developed by Burke (see Appendix E) who determined the additional constraints and input data sets required to properly implement the models.

#### *3.1 Districting*

To use the MSA model for districting, one would input into the model a database of cells from a region representing the populations of different subregions of various shapes (1:1). As was stated in Chapter I, the model will group together those cells which optimize the objective within the constraints of contiguity, compactness, and population deviation (1:2,4). In later revisions of the MSA model, the overall objective is a combination of the following objectives (1:10):

- Minimize the cost of cell acquisition into a subregion,
- Maximize the area of each subregion, and
- Minimize the perimeter of each subregion.

#### *3.2 Single Subregion Allocation (SSA)*

Let each of the  $n$  number of cells of a data matrix be represented by the binary variable  $x_i$ , where  $x_i = 1$  if cell  $i$  is allocated to the subregion and 0 if not. The cost of

each cell,  $c_i$ , could represent many things: the population of each cell in a districting model, the grey value of each pixel in an image, or a distance measurement relative to other cells in any environment. For the purposes of this chapter, they will simply represent a generic cost value. The area of each cell can be represented by  $a_i$ . Finally, the perimeter of a subregion is determined by keeping track of each cell border in the following manner: define  $s_{ij}$  to be the border length between cells  $i$  and  $j$ , and define two new mutually-exclusive binary variables,  $P_{ij}$  and  $N_{ij}$ , which are used to determine 1) if cell border  $s_{ij}$  is also a subregion border, and 2) whether cell  $i$ , cell  $j$ , or both are acquired cells of the subregion.  $P_{ij}$  will equal 1 and  $N_{ij}$  will equal 0 if cell  $i$  is acquired and cell  $j$  is not;  $P_{ij}$  will equal 0 and  $N_{ij}$  will equal 1 if cell  $i$  is not acquired, but cell  $j$  is; finally,  $P_{ij} = N_{ij} = 0$  if both cells  $i$  and  $j$  are acquired or neither is acquired (1:5-6), (6:31-33). The objectives are as follows (1:5-6), (6:32-33):

Minimizing cell acquisition cost:

$$\text{Min } Y_1 = \sum_{i=1}^n c_i x_i \quad (1)$$

Maximizing subregion area:

$$\text{Max } Y_2 = \sum_{i=1}^n a_i x_i \quad (2)$$

Minimizing subregion perimeter:

$$\text{Min } Y_3 = \sum_{i=1}^n \sum_{j \in T_i} s_{ij} (P_{ij} + N_{ij}) \quad (3)$$

where  $T_i$  is the set of all cells adjacent to (immediately above, below, to the right of, and to the left of) cell  $i$ .

To note is the fact that with the Burke implementation of the B&W models, the set of adjacent cells,  $T_i$ , to any cell  $i$  contains those cells that are both logically and physically adjacent to cell  $i$ . Figure 1 shows the four adjacent cells to Cell 1 in a  $5 \times 5$  matrix; Cells 2 and 6 are physically adjacent, and Cells 5 and 21 are logically adjacent. Distinctions between the logically adjacent cells and the physically adjacent cells are made through the use of special input sets included with the input files; they will be explained later.

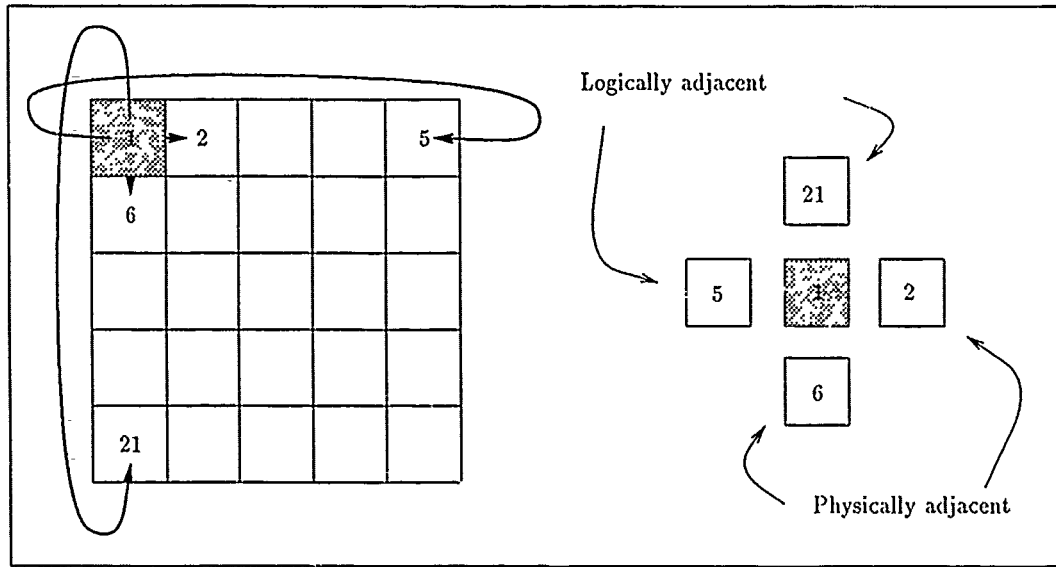


Figure 1. Neighbors of a Matrix Cell.

The three objectives, Equations 1, 2, and 3, can be combined into one objective with weights  $\lambda$  assigned to each one (allowing for the user to give more importance to one objective over another). Letting  $a_i = s_{ij} = 1$  (since each cell will have the same size and border length and thus will not change any results) (8), adding a constraint to set the proper values for  $P_{ij}$  and  $N_{ij}$  (based on the values of  $x_i$  and  $x_j$ ), and adding a constraint to ensure  $P_{ij}$  and  $N_{ij}$  cannot both equal 1 (4), the overall formulation is as follows (6:33-34):

$$\text{Min } Z = \lambda_c \sum_{i=1}^n c_i x_i - \lambda_a \sum_{i=1}^n x_i + \lambda_s \sum_{i=1}^n \sum_{j \in T_i} (P_{ij} + N_{ij}) \quad (4)$$

subject to:

$$x_i - x_j - P_{ij} + N_{ij} = 0 \quad \forall i, j \in T_i \quad (5)$$

$$P_{ij} + N_{ij} \leq 1 \quad \forall i, j \in T_i \quad (6)$$

$$x_i, P_{ij}, N_{ij} \in [0, 1] \quad \forall i, j \in T_i \quad (7)$$

$$\lambda_c + \lambda_a + \lambda_s = 1 \quad (8)$$

A less time consuming and more compact formulation may be written by removing two of the objectives from the objective statement, Equation 4, and allowing

them to be specified as constraints (6:34). If minimizing cost is now the objective, and maximizing area and minimizing perimeter are now constraints, then the user must specify the required subregion area,  $M$ , and twice the required perimeter length,  $L = 2B$ .  $L$  must be twice the required subregion perimeter length (subregion border length  $B$ ) because for each cell  $i$  and adjacent cell  $j$ , their connecting border is counted once for the  $P_{ij}$  and  $N_{ij}$  pair and once again for the  $P_{ji}$  and  $N_{ji}$  pair (8). This new single subregion allocation formulation is as follows (6:34):

$$\text{Min } Z = \sum_{i=1}^n c_i x_i \quad (9)$$

subject to:

$$\sum_{i=1}^n x_i = M \quad (10)$$

$$\sum_{i=1}^n \sum_{j \in T_i} (P_{ij} + N_{ij}) = L = 2B \quad (11)$$

$$x_i - x_j - P_{ij} + N_{ij} = 0 \quad \forall i, j \in T_i \quad (12)$$

$$P_{ij} + N_{ij} \leq 1 \quad \forall i, j \in T_i \quad (13)$$

$$x_i, P_{ij}, N_{ij} \in [0, 1] \quad \forall i, j \in T_i \quad (14)$$

Based now on the subregion area, only certain perimeter lengths are physically possible. The following discussion concerning the range of perimeter lengths allowable comes from the paper by Benabdallah and Wright (1:7-8).

Efficient solutions can be generated using this formulation recognizing that, for any given value of  $M$ —the total area desired—the only feasible values for total external border length  $L$  consist of the even values between  $B_{min}^M$  and  $B_{max}^M$ , where

$$B_{min}^M = 4(\langle \sqrt{M} \rangle) + 2t$$

with:

$$t = \begin{cases} 0 & \text{if } M - (\langle \sqrt{M} \rangle)^2 = 0 \\ 1 & \text{if } M < (\langle \sqrt{M} \rangle)^2 + (\langle \sqrt{M} \rangle) + 1 \\ 2 & \text{if } M > (\langle \sqrt{M} \rangle)^2 + (\langle \sqrt{M} \rangle) + 1 \end{cases}$$

and  $\langle \sqrt{M} \rangle$  = the integer part of  $\sqrt{M}$

and  $B_{max}^M = 4M$  indicating maximum compactness.

The values above for the perimeter lengths could all be used to determine the efficient frontier of solutions given an area of  $M$ . In other words, the models could be run several times with the same subregion area as input, but with incremental changes in the perimeter lengths for each different run. In this way, all the possible minimum-costing subregions could be extracted from the input cost data and compared as far as compactness and cost are concerned. Because two of the objectives of subregion allocation are actually constraints in the final models, this type of several-run approach would be necessary to ensure that the overall optimum solutions for any problem in terms of area, cost, and compactness were obtained.

### 3.3 Single Subregion Allocation Example

6	5	3	4
1	3	2	2
5	4	1	3
2	2	3	1

Figure 2. SSA1:16-Cell Single Subregion Allocation Example Input.

Assume Figure 2 represents 16 cells of a data matrix with the costs as indicated being the costs of acquiring the cells into the subregion. Letting  $M = 2$ , the model will allocate into a single subregion the two adjacent cells whose total cost is the least of any two adjacent cells. Two square, adjacent cells have a total border length of six ( $B = 6$ ); therefore,  $L = 2B = 12$ .

By using the appropriate values for Equations 9-14, the problem SSA1 is formulated as shown below. The solution is shown in Figure 3, and the partial output file, which includes the input file, is located in Appendix B.

6	5	3	4
1	3	②	2
5	4	①	3
2	2	3	1

$z=3$

Figure 3. SSA1:16-Cell Single Subregion Allocation Example Solution.

$$\text{Min } Z = \sum_{i=1}^{16} c_i x_i \quad (15)$$

subject to:

$$\sum_{i=1}^{16} x_i = 2 \quad (16)$$

$$\sum_{i=1}^{16} \sum_{j \in T_i} (P_{ij} + N_{ij}) = 12 \quad (17)$$

$$x_i - x_j - P_{ij} + N_{ij} = 0 \quad \forall i, j \in T_i \quad (18)$$

$$x_i, P_{ij}, N_{ij} \in [0, 1] \quad \forall i, j \in T_i \quad (19)$$

### 3.4 Multiple Subregion Allocation (MSA)

The single subregion allocation model above, Equations 9-14, can be extended to allow for cells to be allocated to one of any number of  $K$  possible subregions.

Let each of the  $n$  number of cells of a data matrix be represented by the binary variable  $x_{ik}$  where  $x_{ik} = 1$  if cell  $i$  is allocated to the subregion  $k \in K$  and 0 if not. The cost of cell acquisition, the area of each cell, and the length of the border separating cell  $i$  and cell  $j$  remain, respectively:  $c_i$ ,  $a_i$ , and  $s_{ij}$ . The mutually-exclusive binary decision variables,  $P_{ijk}$  and  $N_{ijk}$ , are used to determine 1) if cell border  $s_{ij}$  is also a subregion border, and 2) whether cell  $i$ , cell  $j$ , or both are acquired cells of subregion  $k$ .  $P_{ijk}$  will equal 1 and  $N_{ijk}$  will equal 0 if cell  $i$  is acquired in subregion  $k$  and cell  $j$  is not;  $P_{ijk}$  will equal 0 and  $N_{ijk}$  will equal 1 if cell  $i$  is not acquired in subregion  $k$ , but cell  $j$  is; finally,  $P_{ijk} = N_{ijk} = 0$  if both cells  $i$  and  $j$  or neither are acquired in subregion  $k$  (1:9-11). The objectives are as follows (1:9-11):



Minimizing cell acquisition cost:

$$\text{Min } Y_1 = \sum_{i=1}^n c_i x_{ik} \quad \forall k \in K \quad (20)$$

Maximizing subregion area:

$$\text{Max } Y_2 = \sum_{i=1}^n a_i x_{ik} \quad \forall k \in K \quad (21)$$

Minimizing subregion perimeter:

$$\text{Min } Y_3 = \sum_{i=1}^n \sum_{j \in T_i} s_{ij} (P_{ijk} + N_{ijk}) \quad \forall k \in K \quad (22)$$

where  $T_i$  is the set of all cells adjacent to cell  $i$ .

The three objectives, Equations 20–22, can be combined into one objective with relative weights  $W$  assigned to each one. Letting  $a_i = s_{ij} = 1$  as with the previous model (8), adding a constraint to ensure cell  $i$  is assigned to at most one subregion, and adding the appropriate  $P_{ijk}$  and  $N_{ijk}$  constraints (4), the overall formulation is as follows (1:10–12):

$$\text{Min } Z_{tot} = \sum_{k=1}^K Z_k \quad (23)$$

where

$$Z_k = W_c \sum_{i=1}^n c_i x_{ik} - W_a \sum_{i=1}^n x_{ik} + W_s \sum_{i=1}^n \sum_{j \in T_i} (P_{ijk} + N_{ijk}) \quad (24)$$

subject to:

$$\sum_{k=1}^K x_{ik} \leq 1 \quad \forall i \quad (25)$$

$$x_{ik} - x_{jk} - P_{ijk} + N_{ijk} = 0 \quad \forall i, j \in T_i, k \in K \quad (26)$$

$$P_{ijk} + N_{ijk} \leq 1 \quad \forall i, j \in T_i, k \in K \quad (27)$$

$$x_{ik}, P_{ijk}, N_{ijk} \in [0, 1] \quad \forall i, j \in T_i, k \in K \quad (28)$$

$$W_c + W_a + W_s = 1 \quad (29)$$

As with the SSA problem, a less time consuming and more compact formulation may be written by removing two of the objectives from each  $k$  objective statement, Equation 24, and allowing them to be specified as constraints (6:34). If minimizing cost is now the objective, and maximizing area and minimizing perimeter are now constraints, then the user must specify the required subregion area,  $M_k$ , and twice the required perimeter length (8),  $L_k$ , for each subregion  $k$  as with the single subregion model. This new multiple subregion allocation formulation is as follows (1:11-12):

$$\text{Min } Z = \sum_{k=1}^K W_k \sum_{i=1}^n c_i x_{ik} \quad (30)$$

subject to:

$$\sum_{i=1}^n x_{ik} = M_k \quad \forall k \in K \quad (31)$$

$$\sum_{i=1}^n \sum_{j \in T_i} (P_{ijk} + N_{ijk}) = L_k \quad \forall k \in K \quad (32)$$

$$\sum_{k=1}^K x_{ik} \leq 1 \quad \forall i \quad (33)$$

$$x_{ik} - x_{jk} - P_{ijk} + N_{ijk} = 0 \quad \forall i, j \in T_i, k \in K \quad (34)$$

$$P_{ijk} + N_{ijk} \leq 1 \quad \forall i, j \in T_i, k \in K \quad (35)$$

$$x_{ik}, P_{ijk}, N_{ijk} \in [0, 1] \quad \forall i, j \in T_i, k \in K \quad (36)$$

Based now on the subregion area, only certain perimeter lengths for each of the subregions are physically possible. The following discussion concerning the range of perimeter lengths allowable, though not directly taken from the Benabdallah and Wright paper, is an expansion of their coverage of this topic with respect to the single subregion problem (1:7-8) to the more general multiple subregion problem.

As was the case with the subregions of the single subregion problem, the above formulation can be used to generate the efficient solutions so long as the possible feasible perimeter lengths for each subregion are known.

For any given desired subregion of area  $M_k$ , the only feasible values for total external border length  $L_k$  consist of the even values between  $B_{min}^{M_k}$  and  $B_{max}^{M_k}$ , where

$$B_{min}^{M_k} = 4(\langle \sqrt{M_k} \rangle) + 2t$$

with:

$$t = \begin{cases} 0 & \text{if } M_k - (\langle \sqrt{M_k} \rangle)^2 = 0 \\ 1 & \text{if } M_k < (\langle \sqrt{M_k} \rangle)^2 + (\langle \sqrt{M_k} \rangle) + 1 \\ 2 & \text{if } M_k > (\langle \sqrt{M_k} \rangle)^2 + (\langle \sqrt{M_k} \rangle) + 1 \end{cases}$$

and  $\langle \sqrt{M_k} \rangle$  = the integer part of  $\sqrt{M_k}$

and  $B_{max}^{M_k} = 4M$  indicating maximum compactness.

Also, as with the single subregion problem, the models could be run several times with the same subregion areas as input, but with incremental changes in the perimeter lengths for each different run. In this way, all the possible minimum-costing subregions could be extracted from the input cost data and compared as far as compactness and cost are concerned. Recall that this type of several-run approach would be necessary to ensure that the overall optimum solutions for any problem in terms of area, cost, and compactness were obtained since two of the objectives of subregion allocation are actually constraints in the final models

### 3.5 Multiple Subregion Allocation Example

Assume Figure 4 represents 36 cells of a data matrix with the costs as indicated being the costs of acquiring the cells into the subregions. If the problem were to acquire two different square subregions with area of Subregion 1 = 9 and Subregion 2 = 4, then  $M_1 = 9$ ,  $L_1 = (2 \times 12) = 24$ ,  $M_2 = 4$ , and  $L_2 = (2 \times 8) = 16$ .

By using the appropriate values for Equations 30-36 and assuming equal weights for the two cost objectives, the problem MSA1 is formulated as shown below. The solution is illustrated in Figure 5, and the input file is located in Appendix B.

6	5	3	4	3	4
2	5	1	8	5	3
7	7	3	4	9	2
1	3	2	2	6	5
5	4	5	3	7	6
2	3	4	5	8	9

Figure 4. MSA1:36-Cell Multiple Subregion Allocation Example Input.

6	⑤	③	4	3	4
2	⑤	①	8	5	3
7	7	3	4	9	2
①	③	②	2	6	5
⑤	④	⑤	3	7	6
②	③	④	5	8	9

$z=13$

Figure 5. MSA1:36-Cell Multiple Subregion Allocation Example Solution.

$$\text{Min } Z = \sum_{i=1}^{36} c_i(x_{i1} + x_{i2}) \quad (37)$$

subject to:

$$\sum_{i=1}^{36} x_{i1} = M_1 = 9 \quad (38)$$

$$\sum_{i=1}^{36} x_{i2} = M_2 = 4 \quad (39)$$

$$\sum_{i=1}^{36} \sum_{j \in T_i} (P_{ij1} + N_{ij1}) = L_1 = 24 \quad (40)$$

$$\sum_{i=1}^{36} \sum_{j \in T_i} (P_{ij2} + N_{ij2}) = L_2 = 16 \quad (41)$$

$$\sum_{k=1}^2 x_{ik} \leq 1 \quad \forall i \quad (42)$$

$$x_{ik} - x_{jk} - P_{ijk} + N_{ijk} = 0 \quad \forall i, j \in T_i, k \in K \quad (43)$$

$$P_{ijk} + N_{ijk} \leq 1 \quad \forall i, j \in T_i, k \in K \quad (44)$$

$$x_{ik}, P_{ijk}, N_{ijk} \in [0, 1] \quad \forall i, j \in T_i, k \in K \quad (45)$$

### 3.6 A Final Single Subregion Allocation Example

1	1	7	7	1	1
1	1	2	2	1	1
7	2	3	3	2	7
7	2	3	3	2	7
1	1	2	2	1	1
1	1	7	7	1	1

Figure 6. SSA2:36-Cell Single Subregion Allocation Example Input.

Assume Figure 6 represents 36 cells of a data matrix with the costs as indicated being the costs of acquiring the cells into the subregion. If the problem were to

acquire the most compact single subregion with area equal to 16, then  $M = 16$  and  $L = (2 \times 16) = 32$ .

1	1	7	7	1	1
1	①	②	②	①	1
7	②	③	③	②	7
7	②	③	③	②	7
1	①	②	②	①	1
1	1	7	7	1	1

$z=32$

Figure 7. SSA2:36-Cell Single Subregion Allocation Example Solution.

By using the appropriate values for Equations 9-14, the problem SSA2 is formulated as shown below, the solution of which is illustrated in Figure 7.

$$\text{Min } Z = \sum_{i=1}^{36} c_i x_i \quad (46)$$

subject to:

$$\sum_{i=1}^{36} x_i = 16 \quad (47)$$

$$\sum_{i=1}^{36} \sum_{j \in T_i} (P_{ij} + N_{ij}) = 32 \quad (48)$$

$$x_i - x_j - P_{ij} + N_{ij} = 0 \quad \forall i, j \in T_i \quad (49)$$

$$x_i, P_{ij}, N_{ij} \in [0, 1] \quad \forall i, j \in T_i \quad (50)$$

### 3.7 Applicability and Complexity of the B&W Subregion Allocation Models

The B&W model can perform single and multiple subregion allocation of cells and districting as well as enforce user-specified rectangular subregion shape constraints (1), (4). The only real limitations are those arising from problem size since the B&W model relies heavily on many variables and constraints (1), (4), (8).

The most straightforward method which can be used to examine the complexity of the B&W model is to first look at the required data sets, the number of variables,

and the number of equations generated through the Burke implementation of the model (4).

Three different sets, or matrices, are needed for any problem: the initial cost matrix, an adjacency set, and a perimeter set (4). The cost matrix is self explanatory; it contains  $n$  cells. The adjacency set contains  $n$  sets of the four cells adjacent to every cost cell. For corner cells and perimeter cells, a "wrap-around" method of logical adjacency determination is used (4); thus, in the  $4 \times 4$  SSA1 example, Cell 1 had as its logically adjacent cells, Cells 2, 4, 5, and 13. Finally, the perimeter set lists those adjacency cells of corner and border cells which are not physically adjacent to the cells (the "wrap-around" logically adjacent cells) (4); thus, in the SSA1 example, the perimeter set for Cell 1 included Cells 4 and 13. There are  $(2 \times \text{row}) + (2 \times \text{column}) - 4$  of these perimeter sets where *row* denotes the number of cost matrix rows, and *column* denotes the number of cost matrix columns.

The number of variables that are required can be determined through the use of a few simple equations. The major variables in any problem are  $X$ ,  $P$ , and  $N$ . For every MSA problem where  $K$  denotes the number of desired subregions, there are  $nK$  number of  $X$  variables, one for each cell for each subregion. Since the  $P$  and  $N$  variables represent shared borders between any cell and its adjacent neighbor, and since there are  $4 \times K$  possible subregion neighbors (physical as well as logical) for any cell, there are  $4nK$   $P$  variables, and the same number of  $N$  variables. The SSA1 example problem, where  $n = 16$  and  $K = 1$ , had 16  $X$  variables, 64  $P$  variables, and 64  $N$  variables for a total variable count of 145 (including  $Z$ , the objective, as a variable). Problem MSA1, where  $n = 36$  and  $K = 2$  had  $2 \times 36 = 72$   $X$  variables,  $4 \times 36 \times 2 = 288$   $P$  variables, and 288  $N$  variables for a total variable count of 649 (including  $Z$ , the objective, as a variable).

The number of equations required for the Burke implementation of the B&W models can be determined using methods similar to those used for the variable count determination. Each problem has one objective function which counts as an equation. For every subregion  $k$ , there is one equation specifying the size, and another specifying the subregion perimeter length (number of equations =  $2K$ ). For every  $P$  and  $N$  variable combination (every border between a cell and its neighbors for each subregion) there is a constraint to ensure appropriate values for the  $P$  and  $N$  variables (number of equations =  $4nK$ ). For every perimeter cell for every subregion, there is a  $P$  and  $N$  equation for perimeter border values of the  $P$  and  $N$  variables,

and for every physical border between cells for every subregion there is a constraint to ensure the mutual exclusion of the values of the  $P$  or  $N$  variables (together, the number of equations for these two constraints is  $4nK$ ). Finally, for every cell, there is a constraint restricting its allocation to at most one subregion (number of equations =  $n$ ). The total number of equations for problem MSA1 ( $n = 36$ ,  $K = 2$ ), for example, was 1 for the objective,  $2 \times 2 = 4$  for the subregion size and shape constraints,  $2 \times 4 \times 36 \times 2 = 576$  for the three blocks of  $P$  and  $N$  constraints, and 36 for the cell allocation constraints. The total number of constraints for problem SSA2 is 617. Specifically, the determination of the number of variables and equations necessary for the B&W MSA model where  $n$  is the number of cells, and  $K$  is the number of subregions, is as follows: the number of variables equals  $(9nK) + 1$  and the number of equations equals  $(2K) + (8nK) + n + 1$ . These two functions will directly determine the complexity of any B&W MSA problem.

In general, the complexity of any linear programming problem where  $m$  is the number of rows (equations) and  $n$  is the number of columns (variables) is (7):  $mn \times \binom{m+n}{m}$ . Based on this, the relative complexity of each of the example problems is given below. Even though these complexity values appear astronomical in size, the problems themselves were actually solved within seconds of CPU time.

- SSA1 with 147 equations and 145 variables:

$$(147)(145) \times \binom{292}{147}, \text{ which yields } 7.8588 \times 10^{90}.$$

- SSA2 with 334 equations and 331 variables:

$$(334)(331) \times \binom{665}{334}, \text{ which yields } 5.1994 \times 10^{203}.$$

- MSA1 with 617 equations and 649 variables:

$$(617)(649) \times \binom{1266}{617}, \text{ which yields } 7.6142 \times 10^{384}.$$



### 3.8 The Modified Subregion Allocation Models

The following discussion will explain the mathematical development of the modified subregion allocation objectives and constraints as well as discuss such issues as model applicability and complexity. Solutions for the example runs were derived from the Zero/One Optimization Method (ZOOM) solver, a mixed-integer solver, via the use of the General Algebraic Modeling System (GAMS) within a VAX/VMS operating environment (4). Specifics on how to run a GAMS program can be found in Appendix A.

Several changes were made to the B&W model in the hopes of eliminating many of the variables and constraints and allow for a less time-consuming, simpler model which would not only be able to handle larger problems, but would also be easier to model as a network during the later stages of this research. The different philosophies behind the formulations of the B&W model and the Modified model is evident in how they approach the problem of contiguity: the B&W model not only allocates cells to a subregion by restricting the external shape of the subregion (an outside-in approach), but also works from the inside out to join, like the pieces of a jigsaw puzzle, the separate cells together over the entire area of the subregion via the use of the  $P$  and  $N$  binary decision variables; the modified models work predominantly from the outside-in approach. The cell compactness and contiguity constraints deal only with small numbers of cells, not over the entire subregion area. What shall be shown later is that this predominantly outside-in approach is more appropriate when dealing with data already possessing a natural contiguity between cells, such as is the case with the pixels of satellite imagery. This chapter will deal first with the step-by-step development of the modified models and in their applications in a general sense. Chapter V will deal with the applications of the modified models to satellite imagery data.

### 3.9 Modified Single Subregion Allocation (MSSA)

The first major difference between the B&W SSA model (Equations 9–14) and the modified, or MSSA, model is in the representation of the cells themselves. Given a data matrix of  $m$  rows and  $n$  columns, the binary variable  $x_{ij}$ ,  $i \in \{1, \dots, m\}$ ,  $j \in \{1, \dots, n\}$  will be used to represent the cells.  $x_{ij}$  will equal 1 if cell  $ij$  belongs to the subregion, and 0 if not.

The MSSA model is quite different in its constraints with the exception of the overall cost objective and the area constraint. The overall objective, Equation 9, and the area constraint, Equation 10, will have a different representation of the variables but will remain conceptually the same. The cost objective and area constraint are as follows:

$$\text{Min } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (51)$$

subject to:

$$\sum_{i=1}^m \sum_{j=1}^n x_{ij} = M \quad (52)$$

where  $c_{ij}$  is the cost of the cell represented by  $x_{ij}$ , and  $M$  is still used to represent the required area of the desired subregion.

Equations 11, 12, and 13 are all required in the B&W model for contiguity and compactness of the subregion due to the dependence of the model on the  $P_{ij}$  and  $N_{ij}$  binary variables used to keep track of all the cell borders. To reduce the number of equations generated by these constraints, and the sets that need to be declared with them, a different approach based solely on cell neighbors, not cell borders, shall be used to attempt to handle the compactness requirements.

It is necessary at this point to define what is meant by the terms "subregion" and "neighbor." From this point on, a "subregion" means that at least two adjacent cells have been allocated (the allocation of cells to subregions which can have a maximum area of 1 being a rather trivial problem). Those neighbor cells immediately above, below, to the left of, and to the right of a given cell will now be referred to as "first-order neighbors." Those neighbors diagonally above and below, left and right, will now be referred to as "second-order neighbors." Unlike the B&W SSA model, however, the neighbors of each cell do not have to be listed as input sets into the model. Because of the denotation of each cell by its row and column, the neighbor set for any cell,  $x_{ij}$ , is the set of the surrounding eight cells  $\{x_{(i-1,j-1)}, x_{(i-1,j)}, x_{(i-1,j+1)}, x_{(i,j+1)}, x_{(i+1,j+1)}, x_{(i+1,j)}, x_{(i+1,j-1)}, x_{(i,j-1)}\}$  and can be determined real-time during the processing of any computation. The B&W models suffered from border value calculation errors around the matrix edges, and was dealt with through the use of the adjacency and perimeter sets (4). To avoid neighbor value calculation errors around the matrix edges in the modified models, a "frame"

(concept derived through discussions with D. Cameron) will be added to the original image consisting of an extra row of cells at both the top and bottom and an extra column of cells on both the left and right whose costs will equal some large value like 100 or 1000; in this way, the neighbors of every original image cell (now comprising rows 2 to  $(m - 1)$ , and columns 2 to  $(n - 1)$ ) can easily be determined. This extra frame precaution is only needed when dealing with neighbors, and then only some times due to the fact that GAMS generally processes constraints dealing with neighbors correctly even without the frame. More shall be demonstrated about this point later with the Modified Multiple Subregion Allocation example problem. For now, simply note that with any of the following constraints dealing with neighbors, the range of  $i$  may have to be adjusted to  $i \in \{2, \dots, m - 1\}$ , and  $j$  to  $j \in \{2, \dots, n - 1\}$ .

There are many constraints that can be applied to the model to enforce some measure of compactness onto the cells now that the idea of neighbors has been explained. For instance, by requiring each subregion cell to have at least one first-order neighbor, no single cell will ever form a subregion by itself. Besides being a requirement to meet the above definition of "subregion," a constraint such as this could aid in maintaining contiguity, but in itself is not sufficient to enforce contiguity on the entire subregion. If a square subregion were desired, the constraint that each subregion cell have at least one first-order neighbor would not prevent two pairs of non-contiguous cells to be chosen as the subregion cells. Requiring that each subregion cell have at least two first-order neighbors or three total neighbors would force even more compactness and contiguity on the subregion, but there is no guarantee that for a large size subregion, many small  $2 \times 2$  subregions (of area 4) might be formed instead of one large contiguous subregion. Nonetheless, the applicability of the above compactness constraints will be explored and further constraints developed as needed. Examples of typical constraints follow:

The "one-first-order-neighbor-minimum" compactness constraint:

$$\sum_{i=1}^m \sum_{j=1}^n x_{(i-1,j)} + x_{(i+1,j)} + x_{(i,j-1)} + x_{(i,j+1)} - x_{ij} \geq 0 \quad (53)$$

The "two-first-order-neighbors-minimum" compactness constraint:

$$\sum_{i=1}^m \sum_{j=1}^n x_{(i-1,j)} + x_{(i+1,j)} + x_{(i,j-1)} + x_{(i,j+1)} - 2(x_{ij}) \geq 0 \quad (54)$$

Although these equations may at first seem unrelated to their stated compactness goals, they are nothing more than a linear expression stating that “the sum of the neighbor values of a cell minus the cell value itself (or twice the cell value for the ‘2-neighbor’ constraint) must be greater than or equal to zero.” This expression will cause any cell which might be a candidate for allocation to a subregion ( $x_{ij} = 1$ ) to be discarded ( $x_{ij} = 0$ ) if its neighbor sum does not meet the requirement.

Equations 51–54 can be put together with the required binary constraint to formulate the Modified Single Subregion Allocation Model.

$$\text{Min } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (55)$$

subject to:

$$\sum_{i=1}^m \sum_{j=1}^n x_{ij} = M \quad (56)$$

either

$$\sum_{i=1}^m \sum_{j=1}^n x_{(i-1,j)} + x_{(i+1,j)} + x_{(i,j-1)} + x_{(i,j+1)} - x_{ij} \geq 0 \quad (57)$$

or

$$\sum_{i=1}^m \sum_{j=1}^n x_{(i-1,j)} + x_{(i+1,j)} + x_{(i,j-1)} + x_{(i,j+1)} - 2(x_{ij}) \geq 0 \quad (58)$$

$$x_{ij} \in [0, 1], \quad \forall i, j \quad (59)$$

### 3.10 Modified Single Subregion Allocation Example

6	5	3	4
1	3	2	2
5	4	1	3
2	2	3	1

Figure 8. MSSA1:16-Cell Modified Single Subregion Allocation Example Input.

For the solving of problem SSA1 (input data recreated in Figure 8) using the MSSA model, the compactness constraint stipulating that each cell must have at least

one neighbor is used (recall the desired subregion size is  $M = 2$ ; therefore, the cells which will make up the subregion cannot possibly have two first-order neighbors).

6	5	3	4
1	3	②	2
5	4	①	3
2	2	3	1

$z=3$

Figure 9. MSSA1:16-Cell Modified Single Subregion Allocation Example Solution.

The formulation of the problem is shown below, the solution of which is shown in Figure 9, and the partial output file of which is contained in Appendix B.

$$\text{Min } Z = \sum_{i=1}^4 \sum_{j=1}^4 c_{ij} x_{ij} \quad (60)$$

subject to:

$$\sum_{i=1}^4 \sum_{j=1}^4 x_{ij} = 2 \quad (61)$$

$$\sum_{i=1}^4 \sum_{j=1}^4 x_{(i-1,j)} + x_{(i+1,j)} + x_{(i,j-1)} + x_{(i,j+1)} - x_{ij} \geq 0 \quad (62)$$

$$x_{ij} \in [0, 1], \quad \forall i, j \quad (63)$$

### 3.11 Modified Multiple Subregion Allocation (MMSA)

In a manner similar to the B&W Multiple Subregion Allocation model, the MSSA model above, Equations 55-59, can be extended to allow for cells to be allocated to one of any number of  $K$  possible subregions of required area  $M_k$ .

Given a data matrix of  $m$  rows and  $n$  columns, the binary variable  $x_{ijk}$ ,  $i \in \{1, \dots, m\}$ ,  $j \in \{1, \dots, n\}$ ,  $k \in \{1, \dots, K\}$  will be used to represent the cells.  $x_{ijk}$  will equal 1 if cell  $ij$  belongs to subregion  $k$ , and 0 if not. With the addition of a constraint restricting cells to at most one subregion, the entire MSSA model, Equations 55-59, can be changed to reflect the multiple subregion application of the Modified Multiple Subregion Allocation (MMSA) Model (due to compactness problems which shall be discussed shortly, this is not the final formulation):

$$\text{Min } Z = \sum_{i=1}^m \sum_{j=1}^n \sum_{k=1}^K c_{ijk} x_{ijk} \quad (64)$$

subject to:

$$\sum_{i=1}^m \sum_{j=1}^n x_{ijk} = M_k, \forall k \in K \quad (65)$$

either

$$\sum_{i=1}^m \sum_{j=1}^n x_{(i-1,j,k)} + x_{(i+1,j,k)} + x_{(i,j-1,k)} + x_{(i,j+1,k)} - x_{ijk} \geq 0, \forall k \in K \quad (66)$$

or

$$\sum_{i=1}^m \sum_{j=1}^n x_{(i-1,j,k)} + x_{(i+1,j,k)} + x_{(i,j-1,k)} + x_{(i,j+1,k)} - 2(x_{ijk}) \geq 0, \forall k \in K \quad (67)$$

$$\sum_{k=1}^K x_{ijk} \leq 1, \quad \forall i, j \quad (68)$$

$$x_{ijk} \in [0, 1], \quad \forall i, j, k \in K \quad (69)$$

### 3.12 Modified Multiple Subregion Allocation Example

6	5	3	4	3	4
2	5	1	8	5	3
7	7	3	4	9	2
1	3	2	2	6	5
5	4	5	3	7	6
2	3	4	5	8	9


Figure 10. MMSA1:36-Cell Modified Multiple Subregion Allocation Example Input.

Figure 10 represents the same MSA problem input data as that solved by the B&W model. It was decided to use a constraint that stipulated that each cell should

have at least one second-order neighbor in addition to the two-first-order-neighbors constraint.

6	⑤	③	4	3	4
2	⑤	①	8	5	3
7	⑦	③	4	9	2
①	③	②	2	6	5
⑤	④	5	3	7	6
②	③	4	5	8	9

$z=44$

 - s.r. 1


 - s.r. 2

Figure 11. MMSA1:36-Cell Modified Multiple Subregion Allocation Example Solution 1.

The formulation of the problem is shown below, the solution of which is shown in Figure 11.

$$\text{Min } Z = \sum_{i=1}^6 \sum_{j=1}^6 \sum_{k=1}^2 c_{ij} x_{ijk} \quad (70)$$

subject to:

$$\sum_{i=1}^6 \sum_{j=1}^6 x_{ij1} = 9 \quad (71)$$

$$\sum_{i=1}^6 \sum_{j=1}^6 x_{ij2} = 4 \quad (72)$$

$$\sum_{i=1}^6 \sum_{j=1}^6 x_{(i-1,j,k)} + x_{(i+1,j,k)} + x_{(i,j-1,k)} + x_{(i,j+1,k)} - 2(x_{ijk}) \geq 0, k \in [1, 2] \quad (73)$$

$$\sum_{i=1}^6 \sum_{j=1}^6 x_{(i-1,j-1,k)} + x_{(i-1,j+1,k)} + x_{(i+1,j+1,k)} + x_{(i+1,j-1,k)} - x_{ijk} \geq 0, k \in [1, 2] \quad (74)$$

$$\sum_{k=1}^2 x_{ijk} \leq 1, \quad \forall i, j \quad (75)$$

$$x_{ijk} \in [0, 1], \quad \forall i, j, k \in [1, 2] \quad (76)$$

Realizing that the MMSA1 model solution was less compact and yet more expensive than the compact solution, a development which was counter-intuitive, it was decided to try the problem again with a frame of cells around the matrix edges. The allocation of the cells was obviously not registering correctly since the problem being a subregion allocation problem, a multiobjective problem, a less compact solution should have yielded a less expensive solution (or at least not a more expensive solution).

100	100	100	100	100	100	100	100
100	6	5	3	4	3	4	100
100	2	5	1	8	5	3	100
100	7	7	3	4	9	2	100
100	1	3	2	2	6	5	100
100	5	4	5	3	7	6	100
100	2	3	4	5	8	9	100
100	100	100	100	100	100	100	100

Figure 12. MMSA2:36-Cell Modified Multiple Subregion Allocation Example Input.

Figure 12 represents the same MSA problem input data as that solved by the B&W model and the MSSA1 problem but with the addition of a frame of cells. The only compactness constraint used was the two-first-order-neighbors constraint.

The formulation of the problem is shown below, the solution of which is shown in Figure 13, and the input file of which is contained in Appendix B. The MMSA2 model solution was less compact but at least cost the same as the B&W solution. This solution is more acceptable than the MSSA1 solution since it does not violate the "less-compact-more-expensive" erroneous results of the MSSA1 problem.

$$\text{Min } Z = \sum_{i=1}^8 \sum_{j=1}^8 \sum_{k=1}^2 c_{ij} x_{ijk} \quad (77)$$

subject to:

$$\sum_{i=1}^8 \sum_{j=1}^8 x_{ij1} = 9 \quad (78)$$



100	100	100	100	100	100	100	100
100	6	⑤	③	4	3	4	100
100	2	⑤	①	8	5	3	100
100	7	7	3	4	9	2	100
100	1	③	②	②	6	5	100
100	⑤	④	⑤	③	7	6	100
100	②	③	4	5	8	9	100
100	100	100	100	100	100	100	100

$z=43$

- s.r. 1  
 - s.r. 2

Figure 13. MMSA2:36-Cell Modified Multiple Subregion Allocation Example Solution 2.

$$\sum_{i=1}^8 \sum_{j=1}^8 x_{ij2} = 4 \quad (79)$$

$$\sum_{i=2}^7 \sum_{j=2}^7 x_{(i-1,j,k)} + x_{(i+1,j,k)} + x_{(i,j-1,k)} + x_{(i,j+1,k)} - 2(x_{ijk}) \geq 0, k \in [1, 2] \quad (80)$$

$$\sum_{k=1}^2 x_{ijk} \leq 1, \quad \forall i, j \quad (81)$$

$$x_{ijk} \in [0, 1], \quad \forall i, j, k \in [1, 2] \quad (82)$$

In an attempt to enforce more compactness on the subregions, another attempt was made at the MMSA1 problem, input shown in Figure 12. The one-second-order-neighbor compactness constraint was added to the model. The formulation of Problem MMSA3 is shown below, the solution of which is shown in Figure 14.

$$\text{Min } Z = \sum_{i=1}^8 \sum_{j=1}^8 \sum_{k=1}^2 c_{ij} x_{ijk} \quad (83)$$

subject to:

$$\sum_{i=1}^8 \sum_{j=1}^8 x_{ij1} = 9 \quad (84)$$

100	100	100	100	100	100	100	100
100	⑥	⑤	3	4	3	4	100
100	②	⑤	①	8	5	3	100
100	7	7	③	④	9	2	100
100	①	③	②	②	6	5	100
100	⑤	④	5	3	7	6	100
100	2	3	4	5	8	9	100
100	100	100	100	100	100	100	100

$z=43$

- s.r. 1

- s.r. 2

Figure 14. MMSA3:36-Cell Modified Multiple Subregion Allocation Example Solution 3.

$$\sum_{i=1}^S \sum_{j=1}^S x_{ij2} = 4 \quad (85)$$

$$\sum_{i=2}^7 \sum_{j=2}^7 x_{(i-1,j,k)} + x_{(i+1,j,k)} + x_{(i,j-1,k)} + x_{(i,j+1,k)} - 2(x_{ijk}) \geq 0, k \in [1, 2] \quad (86)$$

$$\sum_{i=2}^7 \sum_{j=2}^7 x_{(i-1,j-1,k)} + x_{(i-1,j+1,k)} + x_{(i+1,j+1,k)} + x_{(i+1,j-1,k)} - x_{ijk} \geq 0, k \in [1, 2] \quad (87)$$

$$\sum_{k=1}^2 x_{ijk} \leq 1, \quad \forall i, j \quad (88)$$

$$x_{ijk} \in [0, 1], \quad \forall i, j, k \in [1, 2] \quad (89)$$

What has become apparent is that the compactness constraints used so far are not sufficient, in and of themselves, to enforce compactness on the entire subregion. This development, as well as the lack of a true subregion-wide contiguity constraint mentioned earlier, prompted the exploration into the creation of yet other compactness constraints.

The obvious choice for an additional compactness constraint would be one that enforces some user-defined shape requirements on the subregions in a manner analogous to the way in which the B&W model enforces shape constraints based on subregion perimeters and cell borders, but, of course, doing so without the  $P$  and

$N$  variables and constraints. The shape constraint would have to dictate the height and width of the desired subregion. Looking at Figure 11, one can mentally collapse all the rows down upon one another and see that the size-9 subregion, Subregion 2, is three columns wide. Crunching the columns in upon each other reveals that the subregion is four rows high. While the overall width is acceptable, the height is not; thus the apparent loss in compactness for the subregion.

Expanding on constraints developed by Benabdallah and Wright (1:13), two additional binary variables will be used in the problem:  $R_{ik}$  and  $L_{jk}$ , representing the matrix rows and columns, respectively, which contain cells in subregion  $k$ . Two other variables are used to store the user-defined width and height for each subregion,  $W_k$  and  $H_k$ , respectively. The basic constraints state that if a row (column) does in fact contain cells of a subregion, then the number of cells must equal the width (height) of the subregion to which they belong: these constraints are stating the maximum number of rows and columns that can contain cells of the subregion. Two pairs of additional constraints are needed to control the overall shape, and thus compactness, of the subregions:

$$\sum_{j=1}^n x_{ijk} - (W_k)(R_{ik}) = 0, \quad \forall i, k \in K \quad (90)$$

$$\sum_{i=1}^m R_{ik} - H_k = 0, \quad \forall k \in K \quad (91)$$

$$\sum_{i=1}^m x_{ijk} - (H_k)(L_{jk}) = 0, \quad \forall j, k \in K \quad (92)$$

$$\sum_{j=1}^n L_{jk} - W_k = 0, \quad \forall k \in K \quad (93)$$

The formulation of the final Modified Multiple Subregion Allocation problem: (MMSA4), adding Equations 90-93 to Equations 64-69, is shown below. The input used is the was the original input as shown in Figure 8: no frame was used for this formulation. The solution to the problem is shown in Figure 15, and the partial output file is contained in Appendix B.

$$\text{Min } Z = \sum_{i=1}^6 \sum_{j=1}^6 \sum_{k=1}^2 c_{ij} x_{ijk} \quad (94)$$

subject to:

$$\sum_{i=1}^6 \sum_{j=1}^6 x_{ij1} = 9 \quad (95)$$

$$\sum_{i=1}^6 \sum_{j=1}^6 x_{ij2} = 4 \quad (96)$$

$$\sum_{i=1}^6 \sum_{j=1}^6 x_{(i-1,j,k)} + x_{(i+1,j,k)} + x_{(i,j-1,k)} + x_{(i,j+1,k)} - 2(x_{ijk}) \geq 0, k \in [1, 2] \quad (97)$$

$$\sum_{k=1}^2 x_{ijk} \leq 1, \quad \forall i, j \quad (98)$$

$$\sum_{j=1}^6 x_{ij1} - 3(R_{i1}) = 0, \quad \forall i \quad (99)$$

$$\sum_{j=1}^6 x_{ij2} - 2(R_{i2}) = 0, \quad \forall i \quad (100)$$

$$\sum_{i=1}^6 R_{i1} - 3 = 0 \quad (101)$$

$$\sum_{i=1}^6 R_{i2} - 2 = 0 \quad (102)$$

$$\sum_{i=1}^6 x_{ij1} - 3(L_{j1}) = 0, \quad \forall j \quad (103)$$

$$\sum_{i=1}^6 x_{ij2} - 2(L_{j2}) = 0, \quad \forall j \quad (104)$$

$$\sum_{j=1}^6 L_{j1} - 3 = 0 \quad (105)$$

$$\sum_{j=1}^6 L_{j2} - 2 = 0 \quad (106)$$

$$x_{ijk}, R_{ik}, L_{jk} \in [0, 1], \quad \forall i, j, k \in [1, 2] \quad (107)$$

6	⑤	③	4	3	4
2	⑤	①	8	5	3
7	7	3	4	9	2
①	③	②	2	6	5
⑤	④	⑤	3	7	6
②	③	④	5	8	9

$z=43$

Figure 15. MMSA4:36-Cell Modified Multiple Subregion Allocation Example Solution 4.

### 3.13 A Final Modified Single Subregion Allocation Example

1	1	7	7	1	1
1	1	2	2	1	1
7	2	3	3	2	7
7	2	3	3	2	7
1	1	2	2	1	1
1	1	7	7	1	1

Figure 16. MSSA2:36-Cell Single Subregion Allocation Example Input.

Assume Figure 16 represents 36 cells of a data matrix with the cost as indicated being the costs of acquiring the cells into the subregion. If the problem were to acquire the most compact single subregion with area equal to 16, then  $M = 16$ ,  $H = 4$ ,  $W = 4$ , and thus the maximum number of rows and the maximum number of columns which could contain cells of the subregion would also be equal to 4 ( $R_i = 4$ ,  $L_j = 4$ ).

By using the appropriate values for Equations 64-69, and Equations 90-93, and letting  $K = 1$ , the problem MSSA2 is formulated as shown below, the solution of which is illustrated in Figure 17.

$$\text{Min } Z = \sum_{i=1}^6 \sum_{j=1}^6 c_{ij} x_{ij} \quad (108)$$

subject to:

$$\sum_{i=1}^6 \sum_{j=1}^6 x_{ij} = 16 \quad (109)$$

$$\sum_{i=1}^6 \sum_{j=1}^6 x_{(i-1,j)} + x_{(i+1,j)} + x_{(i,j-1)} + x_{(i,j+1)} - 2(x_{ij}) \geq 0 \quad (110)$$

$$\sum_{j=1}^6 x_{ij} - 4(R_i) = 0, \quad \forall i \quad (111)$$

$$\sum_{i=1}^6 R_i - 4 = 0 \quad (112)$$

$$\sum_{i=1}^6 x_{ij} - 4(L_j) = 0, \quad \forall j \quad (113)$$

$$\sum_{j=1}^6 L_j - 4 = 0 \quad (114)$$

$$x_{ij}, R_i, L_j \in [0, 1], \quad \forall i, j \quad (115)$$

①	①	7	7	①	①
①	①	2	2	①	①
7	2	3	3	2	7
7	2	3	3	2	7
①	①	2	2	①	①
①	①	7	7	①	①

$z=16$

Figure 17. MSSA2:36-Cell Single Subregion Allocation Example Solution 1.

Obviously, this solution, though meeting the constraints, is not the desired solution (see Figure 7). One final set of constraints, tailored specifically for the subregion shape desired, must be added. These constraints state that for any row to contain cells of the subregion, every row within the height of the desired subregion

size must also contain cells of the subregion; the same holds true for the columns of the subregion with respect to the width. The height range can be specified by the terms  $H_{low}$  and  $H_{high}$ ; the width range by  $W_{low}$  and  $W_{high}$ , where the *low* and *high* subscripts denote the lowest and highest numbered row/column which can contain cells of the subregion given that the current row/column contains cells of the subregion. Specifically,  $H_{low} = i - H_{ik} + 1$ ;  $H_{high} = i + H_{ik} - 1$ ;  $W_{low} = j - W_{jk} + 1$ ; and  $W_{high} = j + W_{jk} - 1$ . These new row and column adjacency constraints, which would be unique for each subregion, follow:

$$\sum_{i=H_{low}}^{H_{high}} R_{ik} - (H_k)(R_{ik}) \geq 0, \quad \forall i, k \in K \quad (116)$$

$$\sum_{j=W_{low}}^{W_{high}} L_{jk} - (W_k)(L_{jk}) \geq 0, \quad \forall j, k \in K \quad (117)$$

1	1	7	7	1	1
1	①	②	②	①	1
7	②	③	③	②	7
7	②	③	③	②	7
1	①	②	②	①	1
1	1	7	7	1	1

z=32

Figure 18. MSSA3:36-Cell Single Subregion Allocation Example Solution 2.

By using the appropriate values for Equations 64-69, Equations 90-93, and Equations 116-117 and letting  $K = 1$ , the problem MSSA3 is formulated as shown below, the solution of which is illustrated in Figure 18, and the partial output file of which is contained in Appendix B.

$$\text{Min } Z = \sum_{i=1}^6 \sum_{j=1}^6 c_{ij} x_{ij} \quad (118)$$

subject to:

$$\sum_{i=1}^6 \sum_{j=1}^6 x_{ij} = 16 \quad (119)$$

$$\sum_{i=1}^6 \sum_{j=1}^6 x_{(i-1,j)} + x_{(i+1,j)} + x_{(i,j-1)} + x_{(i,j+1)} - 2(x_{ij}) \geq 0 \quad (120)$$

$$\sum_{j=1}^6 x_{ij} - 4(R_i) = 0, \quad \forall i \quad (121)$$

$$\sum_{i=1}^6 R_i - 4 = 0 \quad (122)$$

$$\sum_{i=1}^6 x_{ij} - 4(L_j) = 0, \quad \forall j \quad (123)$$

$$\sum_{j=1}^6 L_j - 4 = 0 \quad (124)$$

$$R_{i-3} + R_{i-2} + R_{i-1} + R_{i+1} + R_{i+2} + R_{i+3} - 3(R_i) \geq 0 \quad \forall i \quad (125)$$

$$L_{j-3} + L_{j-2} + L_{j-1} + L_{j+1} + L_{j+2} + L_{j+3} - 3(L_j) \geq 0 \quad \forall j \quad (126)$$

$$x_{ij}, R_i, L_j \in [0, 1], \quad \forall i, j \quad (127)$$

This final set of constraints yielded the correct solution. To note is the fact that the constraint was actually written in the GAMS input file as a constraint to apply on every subregion  $k$ ; of coures, there was only one subregion here. For this approach to work on multiple subregions, yet other binary variables would be needed in order to specify the ranges for the rows and columns of each subregion in order to apply this last set of constraints. Based on the fact that this one set of extra constraints in the MSSA3 final solution resulted in a marked increase in the number of branch-and-bound iterations needed to solve the problem (from 0 in MSSA2 to 24,602 in MSSA3), one can deduce that the number of branch-and-bound iterations would increase even more in a multiple subregion problem. Any savings in computation time would soon be overcome. What is actually needed is a way to apply the compactness criteria, forcing contiguity, on a subregion without unduly burdening the branch-and-bound portion of the solution procedure; possibly a way to take into account the cell borders themselves would be the best approach. This conclusion forces the realization that the constraints and variables found in the B&W model are the most appropriate to use in order to fully apply the compactness and contiguity criteria.



### 3.14 Applicability and Complexity of the Modified Subregion Allocation Models

In the problems which have been tested, the MSSA and MMSA models were more compact and quicker than the B&W models. The modified models have fewer overall constraints and variables. In general, only 1/3 to 1/4 of the equations are needed, and only  $(K \times m \times n) + (K \times m) + (K \times n) + 1$  variables are needed, where  $m$  and  $n$  are the number of rows and columns of the framed data matrix,  $K$  is the number of subregions, and the one extra equation is the objective. See Table 1 for a comparison of some basic features and results of the two different formulations. The major flaw with the modified models is that they do not handle subregion contiguity and overall subregion compactness "properly." In other words, contiguity and compactness are not incorporated into the modified models in as full a treatment as they should be, given that they do refer to the third objective of the multi-objective subregion allocation problem. Contiguity for the subregion as a whole can only be obtained through the use of additional decision variables representing the cell borders; this means that if a full exploration into a subregion allocation problem, including dealing with the compactness and contiguity constraints, were desired, only a model formulated to take into account cell borders (such as the B&W model) could handle those constraints. The modified formulations of subregion allocation would be appropriate to use for those instances in which the cells of interest were already configured in some sort of contiguous manner, such as is the case with spatial analysis of satellite imagery, a case study of which is presented in Chapter V.

Table 1. A Comparison of the B&W and Modified Subregion Allocation Models.

Problem Name	Generation + Execution Times	Iterations: LP/B&B	Number of Eqs	Number of Vars	LP Coefs not 0, $\pm 1$	B&B nodes
SSA1*	2.83 sec	152/435	147	145	16	59
MSSA1*	1.13 sec	14/0	18	17	16	0
MSA1*	8.17 sec	874/15669	617	649	72	256
MMSA1	3.13 sec	191/4353	183	73	144	214
MMSA2	3.06 sec	134/5494	139	129	200	384
MMSA3	3.71 sec	204/10796	211	129	200	353
MMSA4*	2.64 sec	185/9044	139	97	168	203
SSA2*	5.01 sec	355/572	334	331	42	16
MSSA2	1.52 sec	57/0	52	49	48	0
MSSA3*	1.43 sec	70/24602	64	49	60	539

\*indicates correct solution obtained

#### *IV. Network Formulations of the Subregion Allocation Problem*

Linear and binary programming problems can be solved much more efficiently using specialized network programming packages if a network structure is embedded in the problem (7). The first task is to find the inherent network structure within a problem, if such a structure does indeed exist. For the problem of finding the minimum costing subregion of size  $M$  out of a matrix of cells, just such a network structure exists. In fact, many different networks can be developed, each with its own specialized nodes, arcs, and side constraints. After experimenting with the basic rudiments of several different network approaches, the research in this area yielded two formulations which stood out above the rest for one reason or another. This chapter contains the development of these two network formulations, example problems and results, a discussion of the complexity and utility of the networks, and conclusions as to the future direction development should take. All the network problems were solved on a VAX/VMS operating system using the NETFLOW procedure of the statistical and network programming package SAS, which supports the use of networks-with-side-constraints (7). Due to the extreme length of the input files for SAS, Pascal programs were written to automatically generate the SAS input files based on the matrix data files and user inputs. Specific instructions on running these programs can be found in Appendix A. The Pascal programs and the partial output files from selected SAS runs are located in Appendix C.

##### *4.1 The Arca-Wall Network Formulation of the Single Subregion Allocation Problem*

Due to the large number of variables and constraints used by the B&W model to enforce contiguity and compactness, essential criteria of subregion allocation, it was decided to attempt a network formulation that possessed an inherent contiguity among its cells so as to possibly avoid having to specify contiguity (and many compactness) constraints. The subregion allocation problem can be formulated as a min-cost network problem with side constraints (a shortest-path problem in the case of the single subregion allocation problem). The side constraints will be explained later.

Let all the  $n$  number of cells of the cell matrix now be represented by network nodes and referred to henceforth as a "wall." There will exist one wall of nodes,

labeled  $a, b, c, \dots$ , for each cell of the desired subregion. The walls are connected to each other by arcs which connect each cell node of one wall to its neighbors in the succeeding wall. The term "cell" now refers to the total collection of nodes with that cell number (i.e., Cell 12 refers to nodes  $a12, b12, c12, \dots$ ). A source node,  $ss$ , will supply a flow of one unit into the first wall of the network, and a destination node,  $dd$ , will demand a flow of one unit from the last wall of the network. The source node is the tail node for  $n$  arcs, one into each cell node of the first wall; therefore, its one unit of flow is allowed to flow into any cell node (can enter Wall  $a$  at any node). Conversely, the destination node is the head node for another set of  $n$  arcs, one from each cell node of the last wall (there is no cost to exit the last wall). Finally, let the cost of acquiring a cell now be the cost of traversing an arc to one of that cell's nodes. The overall objective of this network is to allow one unit of flow to leave the source node, travel into a node of the first wall, through nodes of the other  $M - 1$  walls, exiting the last wall into the destination node, and finding the minimum costing path in the process. The subregion will be made up of those cells which had flow through one of their nodes. In the context of the manner in which a solution is obtained, this is a network problem; however, because the network is being used to solve a subregion *allocation* problem, different "solutions" are permissible. To explain, the ideal network solution would be one in which one unit of flow flows from the source to Wall  $a$ , to a neighbor in Wall  $b$ , to a neighbor in Wall  $c, \dots$ ; however, a solution in which the sum of all flows into a cell equals one (0.5 flow to Node  $a12$  and a subsequent 0.5 flow into Node  $c12$ ), has the possibility of being an acceptable solution as well. An example of this can be found in the discussion of problem AREAWALL2, below. Figure 19 represents how an Area-Wall network would appear if the subregion allocation problem were to determine the minimum costing subregion of size three from an arbitrary cell matrix of size  $n$ .

The general formulation of a network with side constraints (NSC) problem is (7):

$$\text{Min } wx + cy \quad (128)$$

subject to:

$$Ax = b \quad (129)$$

$$Bx + Cy = b' \quad (130)$$

$$0 \leq x \leq d \quad (131)$$

$$0 \leq y \leq D \quad (132)$$

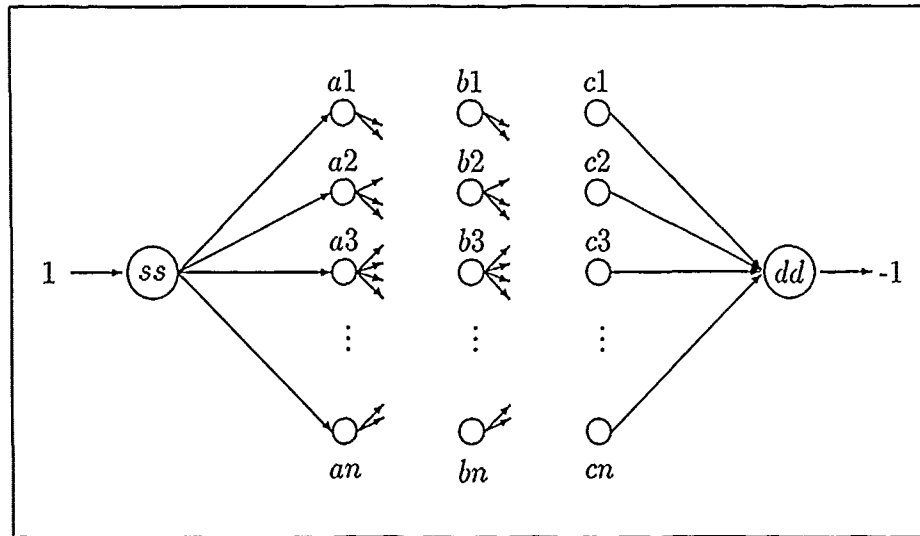


Figure 19. Area-Wall Network Structure for a 3-cell Min-Cost SSA Problem.

where  $x$  and  $y$  are the decision variables of the network and side constraints respectively, both representing arcs in this case;  $w$  and  $c$  are the costs;  $A$  is the node-arc incidence matrix;  $B$  and  $C$  are the arbitrary side constraint matrices;  $b$  is the supply/demand right-hand-side of the  $A$  matrix;  $b'$  is the right-hand-side of the side constraints; and  $d$  and  $D$  are the arc capacities and upper bounds of the variables as applicable (in this case, arc capacities) (7). There are no non-arc variables in the problem (i.e., even the side constraints are constraints in terms of the arcs). At least one set of side constraints are required to ensure that each cell node is visited no more than once. To explain this first set of side constraints, note that each network wall, though containing entirely unique nodes, is in reality, another listing of the original matrix cells preceded by the wall identifier. If a path were to flow from Node  $a1$  to Node  $b2$  to Node  $c1$ , cell 1 would be visited twice; this would not be an acceptable path through the network because a cell cannot be allocated to the subregion more than once which is what the network path would be indicating. To prevent the repeat visiting of nodes, the first set of side constraints ensures that the total flow from a node to any of its neighbors in any wall does not exceed one.

In any given Area-Wall formulation of a single subregion allocation (SSA) problem, the number of nodes, arcs, and side constraints are functions of the initial matrix size and the desired subregion size. Given that  $n$  is the number of cells in a rectangular matrix,  $rows$  is the number of matrix rows,  $cols$  is the number of matrix columns, and  $M$  is the desired subregion size, the functions are:

#nodes = [(# of cells) × (# of network walls)] + (source node) + (destination node)

# nodes =  $nM + 2$

#arcs = (source arcs)+( (total # of cell neighbors) × (# of network wall connections) ) + (destination arcs)

# arcs =  $[(4n - 2rows - 2cols) × (M - 1)] + 2n$

#side constraints = # cells

# set-one side constraints =  $n$

888	888	888	888	888	888
888	6	5	3	4	888
888	1	3	2	2	888
888	5	4	1	3	888
888	2	2	3	1	888
888	888	888	888	888	888

Figure 20. AREAWALL1:16-Cell Single Subregion Allocation Example Input.

*4.1.1 Area-Wall Network Example 1 (AREAWALL1).* Figure 20 represents the original 16-cell SSA problem previously solved via binary programming methods. It is desired to use an NSC formulation to determine the minimum costing subregion of size two. Immediately, it can be determined from the functions above that the number of 1) network nodes =  $(16 \times 2) + 2 = 34$ , 2) network arcs =  $[(4 \times 16) - (2 \times 4) - (2 \times 4)] \times (2 - 1) + 2 \times 16 = 80$ , and 3) side constraints = 16. Recall that the Pascal program, "areawall," used to generate the SAS input file uses an already-generated working file as the matrix data file. This working file matrix contains a frame; thus, when solutions are obtained, the cells that are allocated are sequentially numbered as if the frame cells counted in the overall tally of cells.

The solution is shown in Figure 21, and the input and partial output files are located in Appendix C. The solution can be ascertained from the output by determining which path the flow took through the walls. In this example, the path taken was from the source, *ss*, to Node *a22*, to Node *b16*, to the destination, *dd*. As shall be demonstrated in later example problems, the solutions are generally not so easy to acquire.

888	888	888	888	888	888
888	6	5	3	4	888
888	1	3	②	2	888
888	5	4	①	3	888
888	2	2	3	1	888
888	888	888	888	888	888

$z=3$

Figure 21. AREAWALL1:16-Cell Single Subregion Allocation Example Solution.

4.1.2 *Area-Wall Network Example 2 (AREAWALL2)*. The SSA2 problem previously solved via the binary programming models of the preceding chapter was attempted, but due to the relatively large size of the resulting network (578 nodes, 1872 arcs, and 72 side constraints (includes set-one and set-two side constraints (explained below) ) and the increase in computational effort needed to solve a network with side constraints, the problem was submitted to the long SAS queue. It is not known how many CPU minutes were used, but it is known that it took more than two CPU minutes since the short SAS queue stopped processing the Area-Wall network formulation of problem SSA2 when its two-minute CPU time limit was reached (thus, the submission of the problem to the long SAS queue), and it also took less than 60 CPU minutes since that is the maximum allowed in the long SAS queue. A discussion of the CPU time involved, however, is irrelevant since the problem was not solved completely anyway. The solution without side constraints was 16 (apparently, the four corner cells of each of the four matrix corners); determining this first part of the solution took 474 iterations. Taking into account the side constraints, the SAS network package struggled through nearly 1 million iterations before halting, the only "results" being the following error message (shown verbatim) in the .log file (there was no .lis file) which were repeated 273 times at nearly regular intervals beginning with iteration number 5272 and repeating through iteration number 948393:

ERROR: In iteration [iteration number], a singular working basis matrix was created when a column was replaced by another. The problem may be numerically unstable. ZERO2 may need altering. Try rerunning

using different pricing strategies etc. that may avoid this basis. NOTE: Will assume that the successive column updates have accumulated round-off errors. The working basis matrix will be refactorized.

Having already determined that this particular network formulation would not be practical for any problem size other than "small," inherent difficulties in solving this network, such as is illustrated by the error messages above, came as no surprise.

888	888	888	888	888	888
888	1	1	7	7	888
888	1	1	1	1	888
888	2	2	1	1	888
888	7	2	3	3	888
888	888	888	888	888	888

Figure 22. AREAWALL2:16-Cell Single Subregion Allocation Example Input.

In order to illustrate the kind of solution procedure required to solve the area-wall formulation, another subregion allocation problem was run. Figure 22 represents this new 16-cell problem. It is desired to use an NSC formulation to determine the minimum costing compact subregion of size six. For this problem, in order to try to enforce some measure of compactness on the overall subregion, it would be wise to include compactness constraints dictating that each cell have at least two first-order neighbors. From the conclusions of the preceding chapter, it is understood that this compactness constraint is cell-oriented and not well-suited for enforcing compactness on an entire subregion, but it is about all that can be done in the area-wall network formulation. The application of this compactness constraint is accomplished via a second set of side constraints stipulating that, for every cell, the sum of the flows out of all the neighbors of that cell from any wall minus twice the sum of the flows into that cell in any wall must be greater than or equal to zero; the expression of this constraint is analogous to the binary programming expression of this constraint. There is one of these constraints for each original matrix cell.

As with the previous problem, it can be determined from the functions above that the number of 1) network nodes =  $(16 \times 6) + 2 = 98$ , 2) network arcs =

$[(4 \times 4 \times 4) - (2 \times 4) - (2 \times 4)] \times (6 - 1) + 2 \times 16 = 272$ , 3) set-one side constraints = 16, and, now, 4) set-two side constraints = 16. Again, the solution can be ascertained from the output by determining which path the flow took through the walls (partial output files for the tabulated runs, below, are contained in Appendix C).

Table 2. Flow Results of Problem AREAWALL2, Run Number 1.

To wall a node/flow	To wall b node/flow	To wall c node/flow	To wall d node/flow	To wall e node/flow	To wall f node/flow	To d1 flow
15 / 0.25	9 / 0.25	15 / 0.25	14 / 0.25	8 / 0.25	14 / 0.25	0.25
16 / 0.083	22 / 0.083	16 / 0.416	15 / 0.333	9 / 0.416	8 / 0.416	0.416
			22 / 0.083	23 / 0.083	22 / 0.083	0.083
17 / 0.25	16 / 0.25	22 / 0.25	23 / 0.25	17 / 0.25	16 / 0.25	0.25
23 / 0.333	22 / 0.333	(16 above)				
9 / 0.083	15 / 0.083	9 / 0.083	15 / 0.083	(9 above)		

Minimum cost (Branch & Bound lower bound) = 6.

Unfortunately, in this example, one single path was not taken; rather, the flow was split between many arcs of five original branches. As discussed earlier, the only time a split-flow solution could actually yield the min-cost subregion desired, would be if 1) the number of cells that had any flow at all through them exactly equalled the area of the desired subregion, and 2) the sum of all the flows into each of the cells exactly equalled 1 (i.e., (all flows into Node a12) + (all flows into Node b12) + ... = 1). Since the split-flow solution of problem AREAWALL2 does not fit this description of an allocation solution, the problem must be run again, or more specifically, the "areawall" Pascal program needs to be run again, this time selecting the "set flow on arc(s)" option to set the flow on one or more arcs to 1 in the hopes of driving the solution to a single-path, integral solution or to an acceptable split-flow allocation solution.

What is actually happening is that a branch-and-bound solution is being attempted by adding yet another side constraint. The user must be careful to fully fathom every branch in the branch-and bound tree as he pursues this course of action to ensure that each possible outcome has been fathomed, dominated by a better solution, or determined infeasible (7). In general, the user should pick the most likely branch, using whatever criteria he chooses, and fully fathom that branch until integer solutions are obtained. The criteria used here was to start with the branch that had most of the original flow sent through it. Once an integer solution using



this branch is obtained, that solution will become the new upper bound for any remaining branches. For example, if fully fathoming branch number 1 to an integer solution yields a cost of 10, then 10 is the new upper bound. While fathoming a second branch, if a split-flow solution yields a cost of, say, 12, then one can know with certainty, that 12 is the lower bound for that second branch and that no amount of fathoming further on that second branch will yield a cost lower than 12, or, for that matter, lower than 10. One may then dispense with further fathoming of the second branch, declaring all solutions of that branch "dominated," and move on to fathoming the third branch.

888	888	888	888	888	888
888	1	1	7	7	888
888	1	①	①	①	888
888	2	②	①	①	888
888	7	2	3	3	888
888	888	888	888	888	888

$z=7$

Figure 23. AREAWALL2:16-Cell Single Subregion Allocation Example Solution.

Rather than a discussion at length concerning the branch-and-bound results of the runs covering all five major branches, the solutions concerning the first branching (setting Arc ssa23 to 1) along with pertinent comments about those results have been summarized in Tables 2-6, the partial output files of which are located in Appendix C. The original lower bound of the problem was 6, as that was the lowest costing path through the network. For all the subsequent model runs after run number 1, the "areawall" program was re-run, this time selecting the "situation option" number 2, which allows for the flow on arcs to be set to 1. This re-running of the program and the setting of arc flows was repeated 30 times until all five branches had been fathomed, determined dominated, or found infeasible. Along the way, the optimum, compact, minimum costing subregion of size six was determined. The final solution is illustrated in Figure 23.

All other branches (Arcs ssa15, ssa16, ssa17, and ssa9) and forks of those branches were fathomed in the same manner until all forks of all these branches were determined to be infeasible, dominated, or fathomed to the same solution as that

Table 3. Flow Results of Problem AREAWALL2, Run Number 2.

To wall a node/flow	To wall b node/flow	To wall c node/flow	To wall d node/flow	To wall e node/flow	To wall f node/flow	To d1 flow
23 / 1	22 / 1	16 / 0.5 21 / 0.5	17 / 0.5 15 / 0.5	16 / 0.5 14 / 0.5	17 / 0.5 15 / 0.25 8 / 0.25	0.5 0.25 0.25

Arc ssa23 was set to 1. Minimum cost = 6.5.

Arc a23b22 automatically went to a unit flow.

Fathoming on this branch (Arcs b22c16 and b22c21) must continue.

Table 4. Flow Results of Problem AREAWALL2, Run Number 3.

To wall a node/flow	To wall b node/flow	To wall c node/flow	To wall d node/flow	To wall e node/flow	To wall f node/flow	To d1 flow
23 / 1	22 / 1	16 / 1	17 / 0.5 15 / 0.5	11 / 0.5 9 / 0.5	17 / 0.5 15 / 0.25 8 / 0.25	0.5 0.25 0.25

Arcs ssa23, and b22c16 were set to 1. Minimum cost = 9.

Fathoming on this branch (Arcs c16d17 and c16d15) must continue.

Table 5. Flow Results of Problem AREAWALL2, Run Number 5.

To wall a node/flow	To wall b node/flow	To wall c node/flow	To wall d node/flow	To wall e node/flow	To wall f node/flow	To d1 flow
23 / 1	22 / 1	16 / 1	17 / 1	11 / 1	10 / 1	1

Run number 4 (setting Arc c16d15 to 1) was found to be infeasible.

Arcs ssa23, b22c16, and c16d17 were set to 1. Minimum cost = 18.

This is a feasible solution.

An upper bound of 18 has been established.

Return now to the last fork in the branch and continue fathoming.

Table 6. Flow Results of Problem AREAWALL2, Run Number 6.

To wall a node/flow	To wall b node/flow	To wall c node/flow	To wall d node/flow	To wall e node/flow	To wall f node/flow	To d1 flow
23 / 1	22 / 1	16 / 1	17 / 1	11 / 1	10 / 1	1

Arcs ssa23 and b22c21, were set to 1. Minimum cost = 7.

This is a feasible solution (will eventually prove to be the optimal).

The first attempt on the second fork of branch number 1.

A new upper bound of 7 has been established.

found from run number 6. It should be pointed out that because this is originally an *allocation* problem, multiple optimal solutions to the network always exist, even if there is only one single optimum solution to the allocation problem (i.e., flow from Node a12 to Node b13 results in the same allocation of cells as does flow from Node a13 to Node b12).

Table 7. Flow Results of Problem AREAWALL2, Run Number 8.

To wall a node/flow	To wall b node/flow	To wall c node/flow	To wall d node/flow	To wall e node/flow	To wall f node/flow	To d1 flow
15 / 1	16 / 1	17 / 0.5 22 / 0.5	23 / 0.5 21 / 0.5	17 / 0.5 22 / 0.5	23 / 0.5 21 / 0.5	0.5 0.5

Branch ssa15 is being fathomed.

Arcs ssa15 and a15b16, were set to 1. Minimum cost = 7.

This is a feasible solution (equivalent to the optimal)

because each cell has been allocated a total flow of 1.

Also, as far as attempting to acquire a branch-and-bound solution is concerned, it was mentioned earlier that sometimes it is not necessary to have an integral solution to the Area-Wall network in order to have a feasible solution to an allocation problem; Table 7 illustrates this point.

#### 4.2 The Arca-Wall Network Formulation of the Multiple Subrcgion Allocation Problem

Very little needs to be changed to allow for the network formulation of the multiple subregion allocation problem. What does change is that 1) the flow out of the source equals  $K$ , the number of subregions desired, 2) there is one destination

for each subregion, and 3) the total number of walls depends on the size of the largest subregion desired. There will exist one wall, labeled  $a, b, c, \dots$ , for each cell of the largest desired subregion. The walls are connected to each other by arcs which connect each cell node of one wall to its neighbors in the succeeding wall. A source node,  $ss$ , will supply a flow of  $K$  into the first wall of the network, and one destination node for each subregion ( $d1, d2, d3, \dots$ ) will demand a flow of one unit from the appropriate wall (Wall  $M_k$ ) of the network. The determination of nodes and arcs remains basically the same except that, after having built the network with the largest subregion in mind, each additional subregion will add  $n$  arcs and one destination node to the size of the network. The overall objective of this network (assuming integral flow) is to allow  $K$  units of flow to leave the source node, travel into  $K$  nodes of the first wall, through nodes of the other  $M_k - 1$  walls, exiting the appropriate  $M_k$  wall into the  $d_k$  destination node, finding the minimum costing paths to all  $d_k$  nodes in the process. Figure 24 represents how an Area-Wall network would appear if the multiple subregion allocation problem were to determine the two minimum costing subregions of sizes 5 and 3 from an arbitrary cell matrix of size  $n$ .

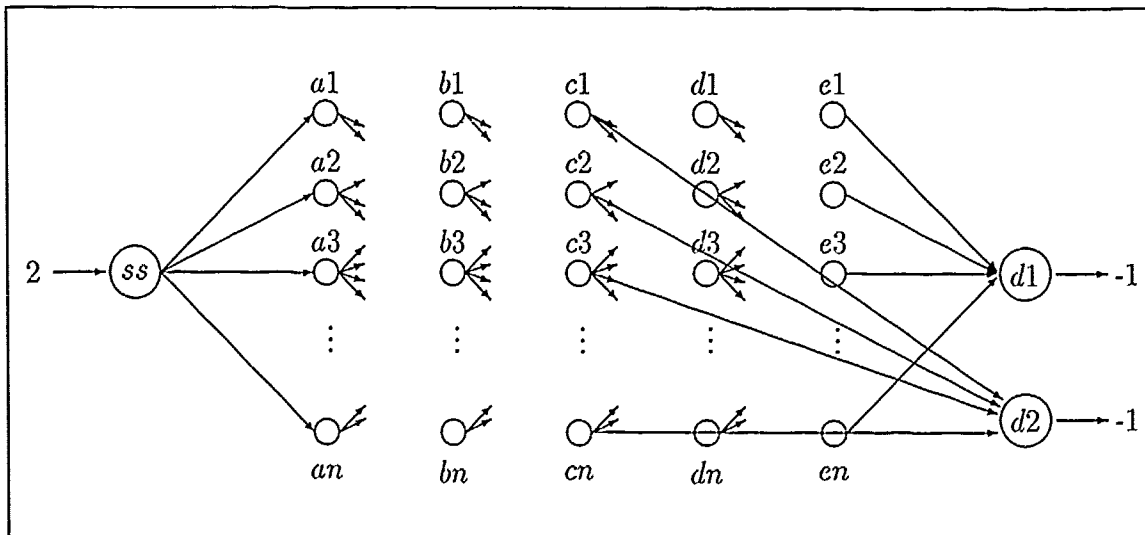


Figure 24. Area-Wall Network Structure for a 5-Cell and 3-Cell Min-Cost MSA Problem.

In any given Area-Wall formulation of a multiple subregion allocation (MSA) problem, the number of nodes, arcs, and side constraints are functions of the initial matrix size, the number of subregions desired, and the size of the largest desired subregion. Given that  $n$  is the number of cells in a rectangular matrix,  $rows$  is the

number of matrix rows, *cols* is the number of matrix columns, *K* is the number of desired subregions, and *M* is the size of the largest desired subregion, the functions are:

#nodes = [(# of cells) × (# of network walls)] + (source node) + (destination nodes)

# nodes =  $nM + 1 + K$

#arcs = (source arcs) + (total # of cell neighbors) × (# of network wall connections) + (destination arcs)

# arcs =  $n + [(4n - 2rows - 2cols)] \times (M - 1) + (Kn)$

#set one and set-two side constraints = twice the # of cells

# set one and set-two side constraints =  $2n$

888	888	888	888	888	888
888	1	1	7	7	888
888	1	1	1	1	888
888	2	2	1	1	888
888	7	2	3	3	888
888	888	888	888	888	888

Figure 25. AREAWALL3:16-Cell Multiple Subregion Allocation Example Input.

Figure 25 represents a 16-cell MSA problem. Other, larger MSA problems were attempted, but only resulted in error statements in the .log files similar to those received when the SSA2 problem was attempted. For this MSA example, it is desired to use an NSC formulation to determine the two minimum costing subregions of sizes six and four (meaning, two destinations are needed; see the partial output file in Appendix C). It can be determined from the functions above that the number of 1) network nodes =  $(16 \times 6) + 1 + 2 = 99$ , 2) network arcs =  $[(4 \times 16) - (2 \times 4) - (2 \times 4)] \times (6 - 1) + 16 + 2 \times 16 = 288$ , and 3) side constraints =  $2 \times 16 = 32$ . Recall that the Pascal program, "areawall," used to generate the SAS input file uses an already-generated working file as the matrix data file. This working file matrix contains a frame; thus, when solutions are obtained, the cells that are allocated are sequentially numbered as if the frame cells counted in the overall tally of cells.

1	2	3	4	5	6
7	(1)	(1)			12
13	(1)	(1)	(1)	(1)	18
19	(.5)	(1)	(1)	(1)	24
25		(.5)			30
31	32	33	34	35	36

Figure 26. AREAWALL3:Allocation Results from Run Number 1.

The solution to the AREAWALL3 problem runs were obtained by using the short SAS queue (the computations took less than two CPU minutes). A different "branch-and-bound" approach was performed on this problem, one that was based on the fact that the true problem here is one of cell allocation and not so much a strict integral network min-cost flow problem. The first step was to examine the output of AREAWALL3, run 1, and determine the amount of allocation of each cell (the sum of the flows through all nodes of each cell). The results of this tally can be found in Figure 26 (see the partial output file in Appendix C) where the frame numbers represent cell numbers, and the numbers in the circles (which represent an allocated cell) represent the amount by which that cell was allocated to one of the subregions (a 1 being full allocation). At this time, it can be determined that Cells 16, 17, 22, and 23 belong to Subregion 2 because of the ideal integer flow from the source to Node *a*17, to Node *b*16, to Node *c*22, to Node *d*23, to Destination *d*2.

At this point, the actual allocation of cells to Subregion 1 had yet to be determined. For this new "branch-and-bound" method, the arc to be set to 1 is that *ss*-to-wall-*a* arc corresponding to a totally allocated cell that also has totally allocated neighbors (the choice was between Arcs *ssa*8, *ssa*9, *ssa*14, *ssa*15, and *ssa*20); in this case, Arc *ssa*8 was chosen. The second run was not as successful as was hoped (reference the partial output files in Appendix C); not only were there still split flow arcs, but now Subregion 2 arcs were split.

Because of these split flow results, the third run arc-setting side constraint contained those original Subregion 2 arcs (to ensure their integer flow was not disrupted) in addition to Arc *a*8*b*14 (which automatically went to an integer flow in run number 2) and Arc *b*14*c*20 (the next fork to fathom on this particular branching)

888	888	888	888	888	888
888	①	①	7	7	888
888	①	①	①	①	888
888	②	②	①	①	888
888	7	2	3	3	888
888	888	888	888	888	888

$z=12$

○

- s.r. 1

⊙

- s.r. 2

Figure 27. AREAWALL3: 16-Cell Multiple Subregion Allocation Example Solution.

(reference Appendix C). The third run proved successful, and this final solution is shown in Figure 27. The partial output files are included in Appendix C. The cell allocation solution can be ascertained from the output by determining which paths the flow took through the walls to the appropriate destinations. Subregion 2 flow has already been described; Subregion 1 consists of Cells 8, 9, 14, 15, 20, and 21 because the flow from the source took the following path (nodes are listed) to the destination d1: a8 to b14 to c15 to d21 to e22 to f23.

#### 4.3 Conclusions of the Area-Wall Network

The major advantage to the Area-Wall network approach is that, given unit flows through the arcs (no split flows), the subregion(s) will be contiguous; this inherent contiguity is due to the fact that flow can proceed from a node in a wall only to a neighbor of that node in the next wall, and so on to the destination. The network alleviates the need for contiguity-enforcing side constraints. The major drawbacks to this formulation are: 1) the addition of side-constraints to the problem causes the SAS solution procedure to take an inordinate amount of time to solve the problem, even with the simplest of problems; and 2) the formulation has no ability to enforce subregion-wide compactness, only being able to enforce compactness and contiguity on a cell-by-cell basis (and, as has been shown with the binary programming models of the previous chapter, there can be no real addressing of the compactness criteria without such an ability).

Generally speaking, network problems are much less complex than their comparable linear and binary programming problems, comparing on the order of complexity

of  $n \times \binom{m+n}{m}$  (where  $n$  is the number of nodes and side constraints and  $m$  is the number of arcs) for networks versus  $mn \times \binom{m+n}{m}$  (where  $m$  and  $n$  are the number of equations and variables) for linear programming (7). Because the node-arc incidence matrix of a network is the comparable construction to the equation-variable matrix of linear and binary programming, the side constraints should also count with the total tally of nodes since they do add to the number of network equations in the node-arc incidence matrix. For the Area-Wall network formulations then, the complexity of solving the NSC model will be based on  $1 + nM + K + 2n$ , where the "1" represents the source node,  $n$  is the number of matrix cells,  $M$  is the number of cells to be allocated to the largest subregion,  $K$  (representing destination nodes) is the number of subregions, and the  $2n$  expression represents the total number of side constraints of an initial problem consisting of both set-one and set-two side constraints. Based on this, the relative complexity of each of the example problems is given below. Even though the complexity values of these network problems is approximately the square root of those of the binary programming problems, the network problems all took longer to solve (minutes versus seconds), and even then only for an initial split-flow solution (with the exception of problem AREAWALL1).

- AREAWALL1 with 34 nodes, 16 side constraints, and 80 arcs:

$$(50)(80) \times \binom{130}{50}, \text{ which yields } 1.188367 \times 10^{10}.$$

- AREAWALL2 with 98 nodes, 32 initial side constraints, and 272 arcs:

$$(130)(272) \times \binom{402}{130}, \text{ which yields } 1.14935 \times 10^{113}.$$

- AREAWALL3 with 99 nodes, 32 initial side constraints, and 288 arcs:

$$(131)(288) \times \binom{419}{131}, \text{ which yields } 1.73833 \times 10^{116}.$$

The network problems took fewer iterations than the binary programming problems (44 iterations for AREAWALL1 versus 592 iterations for SSA1); to note is



the fact that AREAWALL1 was the only problem successfully solved, integer flow, the first time through. Recall that SSA2 was formulated as an Area-Wall network problem and was not solved even after 1 million iterations.

In general, the limitations of networks cannot be changed, but improvements in the overall addressing of the compactness and contiguity criteria in subregion allocation problems can be made. Since one of the conclusions from Chapter III was that cell borders need to be taken into account in order to properly address the compactness and contiguity criteria of subregion allocation, the second network formulation attempts to do just that; it can be considered almost a mirror-image network implementation of the B&W binary programming model.

One interesting phenomenon to note was that, as more arcs had flows set on them in the pursuit of a branch-and-bound solution using the Area-Wall network formulation, the longer the problems took to solve. Since this increase in solution time as the path options were decreased was a counter-intuitive development, a different method of "setting arc flows" in the Area-Wall network formulation should probably have been used. One possible idea is to not set arc-flow via a side constraint, but rather via a structural change in the network itself: the flow from the source could be decreased by the desired flow on the arc to be set, and the head-node of that arc could actually supply the desired flow.

#### *4.4 Version 1 of The Cell-Border Network Formulation of the Single Subregion Allocation Problem*

The idea that nodes will make up "walls" still holds in the "Cell-Border" formulation of the subregion allocation problem. The Cell-Border network consists of three walls of nodes: the  $M$  (matrix cell) wall, the  $B$  (cell border) wall, and the  $C$  (internal border connector) wall. There is one source node,  $S$ , from which flows a quantity equal to four times the total number of desired subregion cells (the total number of cell borders which will be involved). There are three destination nodes:  $DE$ , which demands a quantity equal to the number of external cell borders comprising the subregion perimeter;  $DIR$  which demands a quantity of flow equal to the number of cell borders comprising internal subregion cell connections within rows; and  $DIC$  which demands a quantity of flow equal to the number of cell borders comprising internal subregion cell connections within columns. Arcs of capacity four carry flow from Node  $S$  to Wall  $M$ . Arcs of capacity one carry flow from each

allocated cell to its corresponding cell border nodes in Wall *B* (every border of every allocated cell will be accounted for). The flow from each border node in Wall *B*, carried on arcs of capacity one, either goes directly to Node *DE* or to one of the internal connector nodes of Wall *C* (only those Wall-*B* nodes which are not matrix perimeter nodes have arcs connecting them to connector nodes of Wall *C*). Finally, the Wall *C* nodes have one arc each of capacity two (it takes two adjacent borders to internally connect cells of the subregion) which carry flow into the appropriate internal destination node (*DIR* or *DIC*).

Originally, the cost of cell acquisition was assessed on the *S*-to-Wall-*M* arcs, but because of later developments in Version 2 of the Cell-Border network, this was changed and retroactively applied to Version 1. The cost of acquiring a cell is now assessed on the Wall-*M*-to-Wall-*B* arcs. Because four units of flow must travel from *S* into each allocated cell node of Wall *M*, applying the cost to the *M* cell on each *S*-to-Wall-*M* arc is equivalent to applying the cost to each *M*-to-Wall-*B* arc; either way, the final network solution cost is actually four times the cost of the subregion allocation problem, but that is of no particular consequence. The overall objective of this network is to allow a quantity of flow equal to the total number of desired subregion cell borders (four times the desired number of cells) to leave the source node, travel into cell nodes of Wall *M* (four units of flow into each allocated cell node), through the cells' border nodes and connector nodes of Walls *B* and *C*, respectively, and exit into the appropriate destination (depending on whether the cell borders are subregion external borders, internal row borders, or internal column borders), and finding the minimum costing paths in the process. The subregion will be made up of those cells which have flow through their Wall *M* nodes and through their corresponding Wall *B* and Wall *C* nodes.

Figure 28 shows how a data matrix of cells is conceptually divided into the appropriate nodes for use in the Cell-Border NSC formulation of the subregion allocation problem.

Figure 29 represents how the lower right-hand corner (the three lower right-hand corner cells of the data matrix) of a Cell-Border network, Version 1, would appear for a single subregion allocation problem where the objective were to determine the minimum costing  $2 \times 3$  (two rows by three columns) subregion of size six from an arbitrary cell matrix of size  $n$ . The total number of subregion external cell

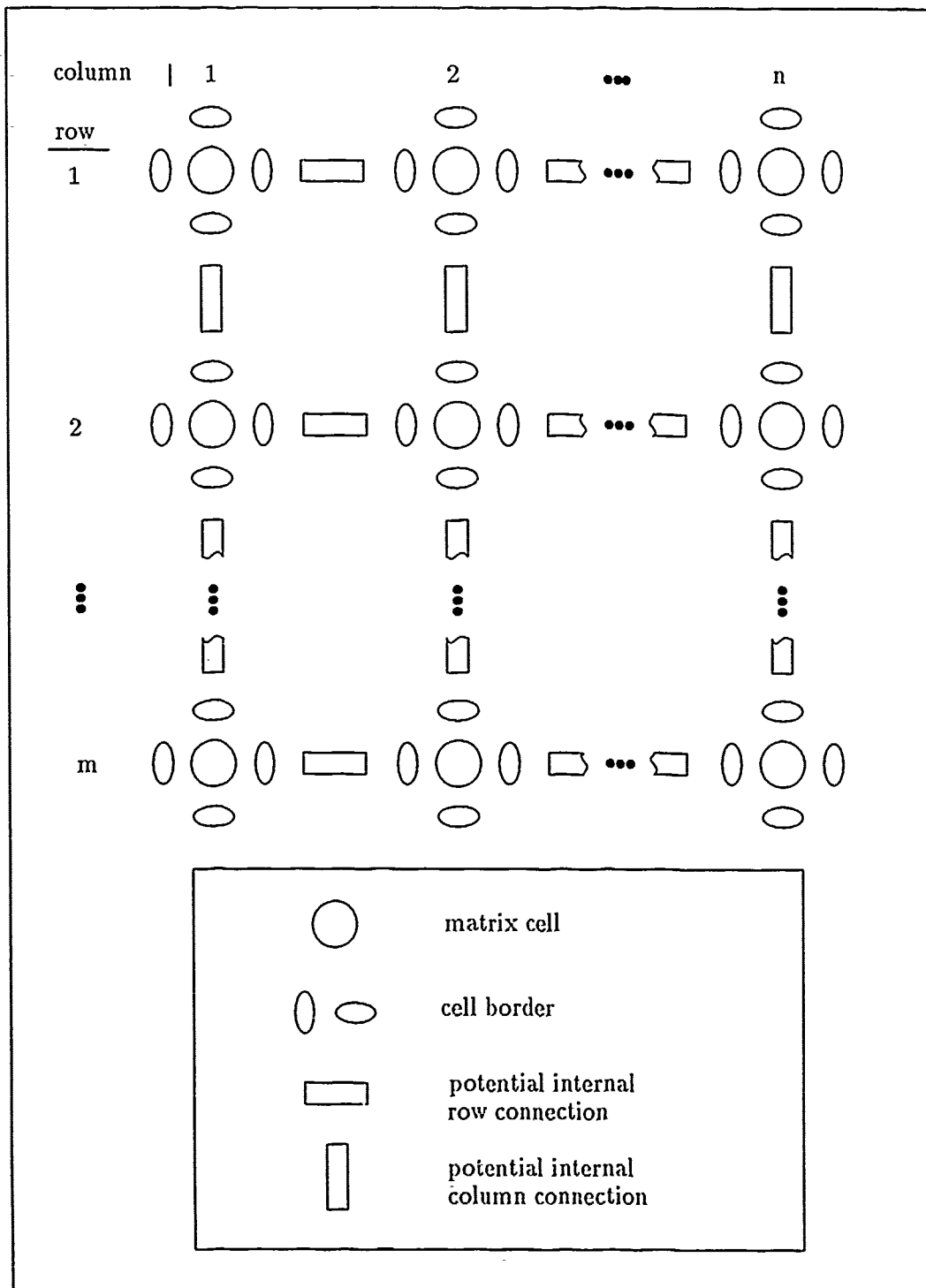


Figure 28. Break-Down of a Data Matrix into its Cell-Border Nodes.

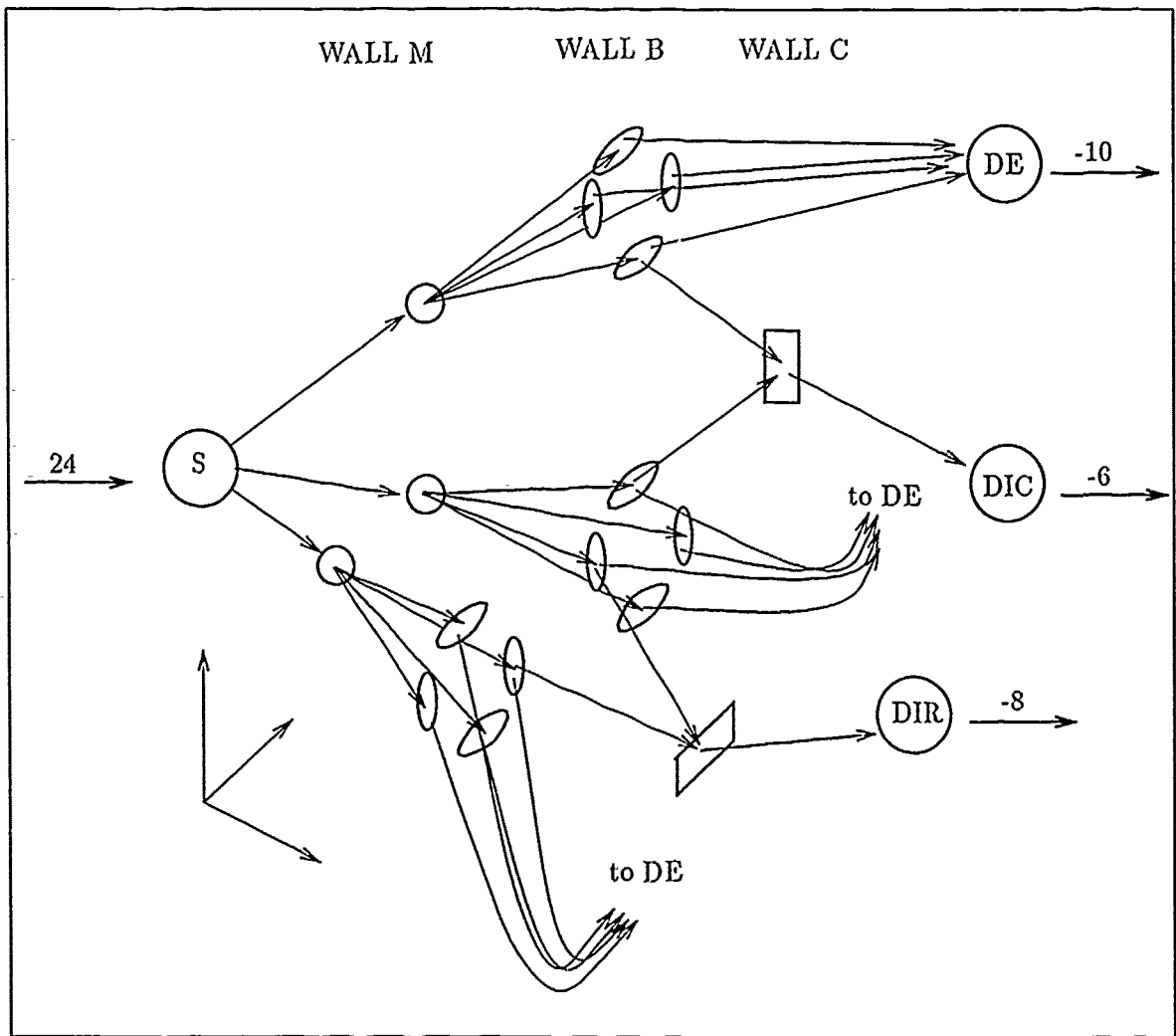


Figure 29. Lower-Right-Hand Corner of the Example Cell-Border Network.

borders equals ten; of internal cell row borders equals eight; of internal cell column borders equals six.

888	888	888	888	888	888
888	6	5	3	4	888
888	1	3	2	2	888
888	5	4	1	3	888
888	2	2	3	1	888
888	888	888	888	888	888

Figure 30. CELLBORDER1: 16-cell Single Subregion Allocation Example Input.

Figure 30 represents the input data matrix of the 16-cell SSA1 problem previously solved via binary programming. It is desired to use the Cell-Border network formulation to determine the minimum costing single subregion of size two (height = 2; width = 1).

The first-run SAS input file contained no side constraints in order to help determine the kinds of side constraints which may be necessary. To note is the fact that a  $2 \times 1$  subregion will have no row connections; therefore, only two destinations are actually used: *DE* and *DIC*, which have demands of 6 and 2, respectively. Run 1 of the CELLBORDER1 problem resulted in four units of flow being allocated to Cell 11, three to Cell 22, and only one to Cell 29 (see the partial output listing in Appendix C). In addition, internal connections (Nodes *c21* and *c20*) had flow on them even though each only had flow on one of their parent borders.

Since two of the Cell-Border formulation goals are to use a connector node only when the two parent borders it connects are also used, and to have all four borders of an allocated cell accounted for, two sets of side constraints seemed necessary. Set-one side constraints will ensure that connector nodes are not used unless the flow into them from both their parent border nodes is the same (i.e., they have to connect something). Set-two side constraints will ensure that the flow across any border of a cell is the same for all borders of that cell. Problem CELLBORDER1, run 2, contained these side constraints (see input listing in Appendix C).

Specifically, set-one side constraints state that the difference between the flows across the parent border nodes of each connector node is zero, and set-two side

888	888	888	888	888	888
888	6	5	3	4	888
888	1	3	②	2	888
888	5	4	①	3	888
888	2	2	3	1	888
888	888	888	888	888	888

$z=12$

Figure 31. CELLBORDER1: 16-Cell Single Subregion Allocation Example Solution.

constraints state that the difference between the flows across every pair-wise combination of a cell's borders and its other two borders is zero. The border difference is taken pair-wise as opposed to single border versus single border in order to cut the number of set-two side constraints from six per cell to three per cell. The solution to run number 2 of the CELLBORDER1 problem is the correct solution and is shown in Figure 31, a partial listing of the output files of which are contained in Appendix C.

In any given Cell-Border formulation of an SSA problem, the number of nodes, arcs, and side constraints are functions of the initial matrix size. Given that  $n$  is the number of cells in a rectangular matrix,  $row$  is the number of matrix rows, and  $col$  is the number of matrix columns, the functions are:

$$\#nodes = (\# \text{ of cells}) + (\# \text{ of cell borders}) + (\# \text{ of connector nodes}) + (\text{source node}) + (\text{destination nodes})$$

$$\# \text{ nodes} = n + 4n + (2n - row - col) + 1 + 3$$

$$= 7n + 4 - row - col$$

$$\#arcs = (\text{source arcs}) + (\text{M to B arcs}) + (\text{B to C arcs}) + (\text{B to DE arcs}) + (\text{C to DIR/C arcs})$$

$$\# \text{ arcs} = n + 4n + (4n - 2row) - 2col + 4n + (2n - row - col)$$

$$= 15n - 3row - 3col$$

$$\#side \text{ constraints} = \# \text{ of connector nodes} + 3 \times (\# \text{ cells})$$

$$\# \text{ set-one and set-two side constraints} = (2n - row - col) + 3n$$

$$= 5n - row - col$$

888	888	888	888	888	888
888	1	1	7	7	888
888	1	1	1	1	888
888	2	2	1	1	888
888	7	2	3	3	888
888	888	888	888	888	888

Figure 32. CELLBORDER2: 16-Cell Single Subregion Allocation Example Input.

Problem AREAWALL2 (input shown in Figure 32) was formulated as a Cell-Border network-with-side constraints problem using the same two sets of side constraints described above for problem CELLBORDER1.

888	888	888	888	888	888
888	①	①	7	7	888
888	①	①	1	1	888
888	②	②	1	1	888
888	7	2	3	3	888
888	888	888	888	888	888

$z=32$

Figure 33. CELLBORDER2A: 3-Row-by-2-Column Single Subregion Allocation Solution.

The problem was solved once for the minimum costing  $3 \times 2$  subregion (problem CELLBORDER2A), and once for the minimum costing  $2 \times 3$  subregion (problem CELLBORDER2B). The partial output files of both runs are located in Appendix C. The solution to problem CELLBORDER2A is shown in Figure 33. Version 1 of the Cell Border network formulation successfully determined the subregion desired on the first run.

Problem CELLBORDER2B did not yield a correct solution on the first run, the cell allocation of which is represented in Figure 34, where a totally allocated cell is shown by a 4, and a partially allocated cell is shown with a 3, 2, or 1 depending on

1	2	3	4	5	6
7	③	③			12
13	③	③	③	③	18
19	①	①	②	②	24
25					30
31	32	33	34	35	36

$z=26$

Figure 34. CELLBORDER2B: Allocation Results From Run Number 1.

the flow through the cell's corresponding Wall  $M$  node from Source  $S$ . The numbers in the frame cells indicate the cell numbers, not a cost value.

The CELLBORDER2B problem indicated that more work on the overall Cell-Border network formulation was needed, but first, actually solving the problem was tried. Arc-setting seemed to be the place to start, and, in fact, the former method of arc-setting was attempted for two different  $S$ -to-Wall- $M$  arcs (Sm14, and Sm9), but the solutions were even more broken up (partially allocated cells) than before, but with cost basically the same (only the partial output files of one of these runs, problem CELLBORDER2B run number 2, setting Arc Sm14 to 4, is contained in Appendix C). Interestingly enough, the cost of the split-flow solution to problem CELLBORDER2B run 2 was 28, the true cost had the correct solution actually been obtained (which would have been to put 4 units of flow into Cells 15, 16, 17, 21, 22, and 23). Because of the fact that one major difference between the Cell-Border formulation and the Area-Wall formulation is that the Cell-Border solution must have more than one arc between walls (must find more than one single path through the network), and because of the inability to significantly effect a change in the solution through setting a source arc, a different "branching-and-bounding", if any is possible, will probably have to be used. Before determining if such a new method were possible, more constraints were attempted.

What seemed to be lacking was a way of pushing the initial flow into a Wall- $M$  cell up to a quantity of four units through the use of specialized side constraints. Based on previous work with non-linear programming, it was determined that this dilemma was analogous to attempting to force "binariness" on a positive variable (i.e., attempting to force, via constraints, a positive variable  $X \in [0..1]$  to either



0 or 1, a task which cannot be done in non-linear programming). Never-the-less, attempts were made, but, as expected, they proved unsuccessful.

A compactness set of side constraints (set three), analogous to the binary programming compactness/contiguity constraints of Equation 54, was added to the network to try to ensure that no cells were allocated without at least two other first-order neighbor cells being allocated by the same amount (ideally by a quantity of four). These constraints state that, for every source arc to a Wall- $M$  cell node, at least two or more other source arcs have to carry the same amount of flow into at least two first-order "neighbors" of the Wall- $M$  cell node.

1	2	3	4	5	6
7	②	②			12
13	③	③	③	③	18
19	①	①	③	③	24
25					30
31	32	33	34	35	36

$z=26$

Figure 35. CELLBORDER2B: Allocation Results From Run Number 3.

Implementing this change in the network and running the problem again for the official third run yields the allocation solution shown in Figure 35, the partial output files of which are located in Appendix C. As in Figure 34, the circles represent at least a partial allocation of a cell, the degree of which being the number within the circle (a 4 meaning totally allocated).

The added compactness constraints applied to the  $S$ -to-Wall- $M$  arcs changed the solution, but not in the way desired. This is due to the fact that, like the non-linear programming models, the Cell-Border network is not dealing strictly with binary variables, and, unfortunately, the incorporated compactness constraints only acquire the desired results (enforcing a minimum tally of first-order neighbors) for binary cases. (If the flow into Wall  $M$  could be forced to either 0 or 4, then the constraints would be effective; what is actually needed is flow that can be considered either on or off, not partial or split).

What is possibly of more interest now is the fact that neighbor cells have been apparently allocated without their shared connector nodes having flow through

them. Even though the border node parents of each connector node do contribute equally, their "equal" contribution in some cases is zero. One of the premises of the Cell-Border network is that if two adjacent cells are allocated to the subregion, the flow through their shared connector node must equal two, one unit of flow for each adjacent border, and it is that flow that counts for an internal row or internal column connection. A set of side constraints to enforce the proper use of the connector nodes was needed.

A constraint relating the flow into a cell with the flow out of its connector nodes may at first appear to be the kind of constraint needed since at least one connector node would need to be used; however, not knowing prior to the allocation if the cell is a subregion corner cell, perimeter cell, or internal cell, it would not be known if two, three, or even all four connector nodes would need to be used. A constraint such as this would result in an inequality statement such as "the flow from the sum of a cell's connector nodes must be greater than 4" (implying that 2 connector nodes at a minimum need to be used for every allocated cell), when what would be more useful and help drive the flows to the integer values desired would be an equality constraint specifying the exact flow through connector nodes based on the neighbors of the cells (the exact number of connector nodes needed).

It seemed at first that this second, equality connector-node constraint could be accomplished via a set of side constraints relating each connector node to its two parent cells rather than each cell to all its possible connector nodes. The constraint that could be applied to each connector node would be one that stated that the flow out of each connector node must equal one-fourth the flow into its two parent cells; specifically, 4 times the flow out of each connector node must equal the flow into the connector node's two parent cells. Unfortunately, it is desired to apply this constraint only to those cells actually allocated as neighbors, a delineation of which cannot be made. The reason why this constraint cannot be used is that all side constraints must apply to, and be true for, all their respective arc variables whether the cells are allocated with neighbors or not. Specifically, a cell can be allocated and have only one neighbor. Attempting to apply a constraint stating that the flow out of every connector must equal one-fourth the flow into the connector node's parents would fail for the allocated cell's three perimeter borders with no neighbor.

Because of this restriction for what appeared at first to be the more encompassing equality "relate Wall-M-to-Wall-C" constraint, the original inequality "relate

1	2	3	4	5	6
7	②	②			12
13	③	③	③	③	18
19	①	①	③	③	24
25					30
31	32	33	34	35	36

$z=26$

Figure 36. CELLBORDER2B: Allocation Results From Run Numbers 4 and 5.

Wall-*M*-to-Wall-*C*" constraint (stating that at least two connector nodes must be used for every allocated cell-side constraint set number four) was applied, once with the first two sets of side constraints (run 4), and once with all three of the other sets of side constraints (run 5). Both runs (4 and 5) obtained the same cell allocation solution as run number 3. Only the partial output file of run number 5 is located in Appendix C.

#### 4.5 Version 2 of The Cell-Border Network Formulation of the Single Subregion Allocation Problem

Having been unsuccessful at solving problem CELLBORDER2B even after logically deriving and applying several different sets of constraints, the true nature of the major underlying problem became apparent. The two parts of the underlying problem are 1) it is not possible to force four units of flow (versus three, two, or one) from the source into any Wall-*M* node, and 2) it is not possible to keep flow from splitting (.25, .5, .75) at any node where two or more arcs depart (so even when four units of flow do get passed through to Wall *M*, it is not possible to keep the flow from splitting between *B*-to-*C* arcs and *B*-to-*DE* arcs further in the network).

Both of these problems are due to the fact that one cannot specify in the SAS files that an integral solution is required. One small example will illustrate the different ways in which a constraint is met in an ideal network package which does enforce integrality flow versus a network package which does not (SAS). Assume the Cell-Border network model solves a problem and determines that some of the cells should receive a flow of three units and that those three units of flow pass through Wall *B* using only three border nodes of the cell, each passing along one unit of

flow. Set-two side constraints state that the flow through all borders of a cell must be the same. A network package which enforces integral flows has only two choices: increase the flow into the cell to four units, and thus every border carries a flow of one, or remove all flow from the cell. SAS, on the other hand, has a wide selection of constraint-meeting options, the most popular of which (based on the example problem solutions obtained in this research) is to split the three units of flow equally among the four borders by passing three-fourths of a unit of flow through each border node; thus, constraint after constraint can be met without acquiring a proper cell allocation solution; and, in some cases, as more integral-flow-enforcing constraints are added, the solution actually degenerates further (the example above in which the one unit of flow on each of three borders degenerates into the three-fourths of a unit of flow on each of four borders is a good illustration).

Realizing that nothing could change the inherent solution algorithms used by the SAS networks-with-side-constraints package, Version 2 of the Cell-Border network formulation was created in the hopes of at least discouraging many of the so-called constraint-meeting options available to SAS.

The purpose of the first change was to address part one of the underlying problem: the failure to obtain a flow of four units into allocated cells. Realizing that four units of flow per allocated cell do in fact exist, but that one or two units may pass through other, nearby cells, it was decided to increase the flow into Wall  $M$  (from the source) from four units of flow per desired subregion cell to four units of flow per matrix cell. With  $4n$  units of flow being supplied by the source, and one of the  $n$  arcs of capacity entering each cell node of Wall  $M$ , the network has no alternative but to allocate exactly four units of flow to every cell. Flow cannot be removed from a potential subregion cell and sent through some other cell because no other cells have any capacity left over. Four units of flow into each allocated cell is now assured, as well as unit flows into each cell border. A fourth destination node (Node  $DN$ ) was added in order to have a sink for all the extra flow. The demand for Node  $DN$  is equal to the total number of non-allocated cell borders. This first major change of the Cell-Border network formulation can be summed up in one sentence: every border of every matrix cell will be accounted for.

It should be noted that the set-two side constraints (every Wall- $M$ -to-Wall- $B$  arc of an allocated cell must carry equal flow) are unnecessary in the Version 2 formulation since every cell border has no choice but to carry a flow of one unit.

Because every cell does have four units of flow, set-three side constraints (minimum of two neighbors) are also unnecessary, and set-four side constraints (cells with flow must have flow on at least two connectors) are not possible. However, the set-two side constraints are replaced with a new fifth set of side constraints stating that, for each cell, every Wall-*B*-to-Destination-*DN* arc must be the same (i.e., in order for a cell to be counted as a “non-allocated” cell, every one of its four borders must carry the cell’s flow to the *DN* destination).

1	2	3	4	5	6
7	②	②			12
13	③	③	③	③	18
19	①	①	③	③	24
25					30
31	32	33	34	35	36

$z=26$

Figure 37. CELLBORDER2B: Allocation Results From Run Number 6.

Problem CELLBORDER2B was attempted again (run number 6) incorporating the Version 2 changes and including the set-one side constraints (see the input listing in Appendix C). The allocation results are shown in Figure 37, where circles represent at least a partial allocation of the cell based on the sum of that cell’s flows to the external subregion border destination (*DE*) and the internal border destinations (*DIR* and *DIC*). The partial output files are included in Appendix C. The solution remained unchanged, unfortunately. Since the splitting of flows from the border nodes could not be prevented through the network construction itself, a means of relating those arcs leading away from the border nodes to one another seemed necessary.

As the Cell-Border network stands now, there is no distinction made between a non-allocated border and any other non-allocated border (in Version 2); there is also no distinction made between an allocated subregion border that is a perimeter border and one that is an internal connecting border (in either Version 1 or Version 2). The second major change of Version 2 of the Cell-Border network attempts to try to relate non-allocated cell borders to allocated perimeter cell borders to allocated internal connecting cell borders. The relationship between the three classifications of

borders and the objectives they are attempting to meet are as follows: it is better to allocate cheaper borders (cells) than more expensive borders (to minimize cost); it is better to "non-allocate" expensive borders (cells) than cheaper borders (again, to minimize cost); it is better to connect as many allocated borders as possible than to have many perimeter borders (increase compactness); and, finally, it is better that the more expensive allocated cells have the most connections: a fourth objective that can be added to the basic subregion allocation problem (a sideline objective dealing with the distribution of the cost within the allocated subregion: if one were allocating population cells, he would want the bulk of the population in the middle of the subregion as opposed to the perimeter of the subregion so as to minimize the future costs of laying cables, building water pipes, constructing roads, and collecting garbage, for example). The two main underlying constraints that must be maintained when assessing the new arc costs are that 1) at no point, should the allocation of an expensive cell cost less than the allocation of a cheaper cell, and 2) at no point should the allocation of any cell be cheaper than not allocating that cell.

To meet these border-relationship standards, the cost assessment of the network in both the areas of allocation and non-allocation will be changed. Allocation costs will now be assessed as follows: the basic cost of cell allocation will be assessed on the Wall-*M*-to-Wall-*B* arcs; no cost will be assessed on the Wall-*B*-to-Node-*DE* arcs; finally, to "reward" cell connections, the Wall-*B*-to-Wall-*C* arcs will all have a "cost" of -1. An example of all possible allocation methods for a cell of cost 10 is shown in Figure 38; note that the first main underlying constraint, that a cell which costs less than some other cell must always cost less than the other cell regardless of how either is allocated, is met (the least expensive way to allocate a cell of cost 10 is still costlier than the most expensive way to allocate a cell of cost 9).

Non-allocation costs, the negative value of twice the original cell cost, shall be assessed on the Wall-*B*-to-Node-*DN* arcs in order to reflect the cells' relative values as far as *not* allocating them is concerned (it being more desirable to not allocate expensive cells rather than to not allocate cheap cells, and, if at all possible, to not allocate any cells). Figure 39 illustrates how the non-allocation costs are assessed in Version 2 of the Cell-Border network.

The additional objective mentioned previously, centralizing cost distribution (that more expensive cells be located near the center of the subregion rather than

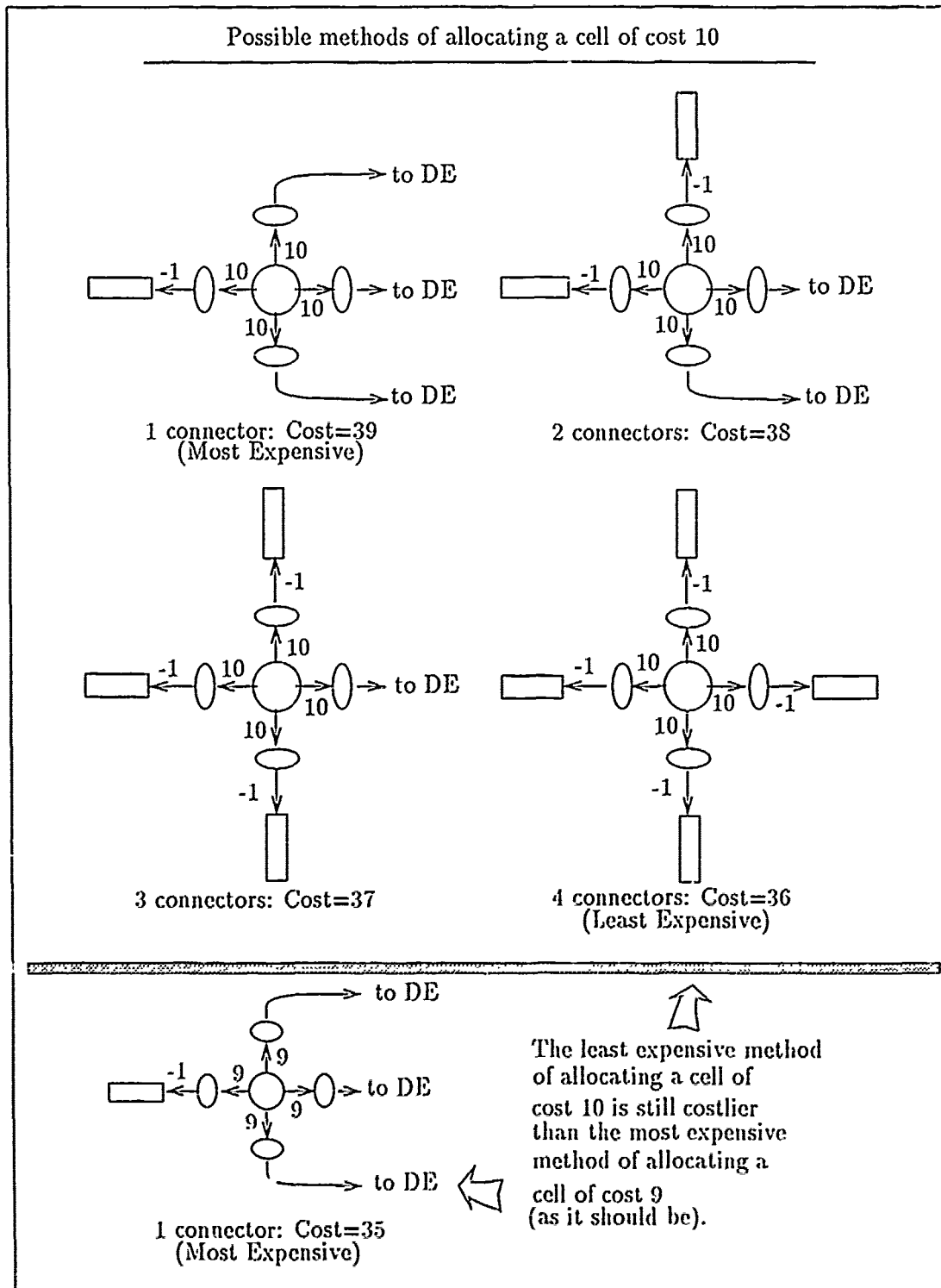


Figure 38. Allocation Cost Assessment of the Cell-Border Network Version 2.

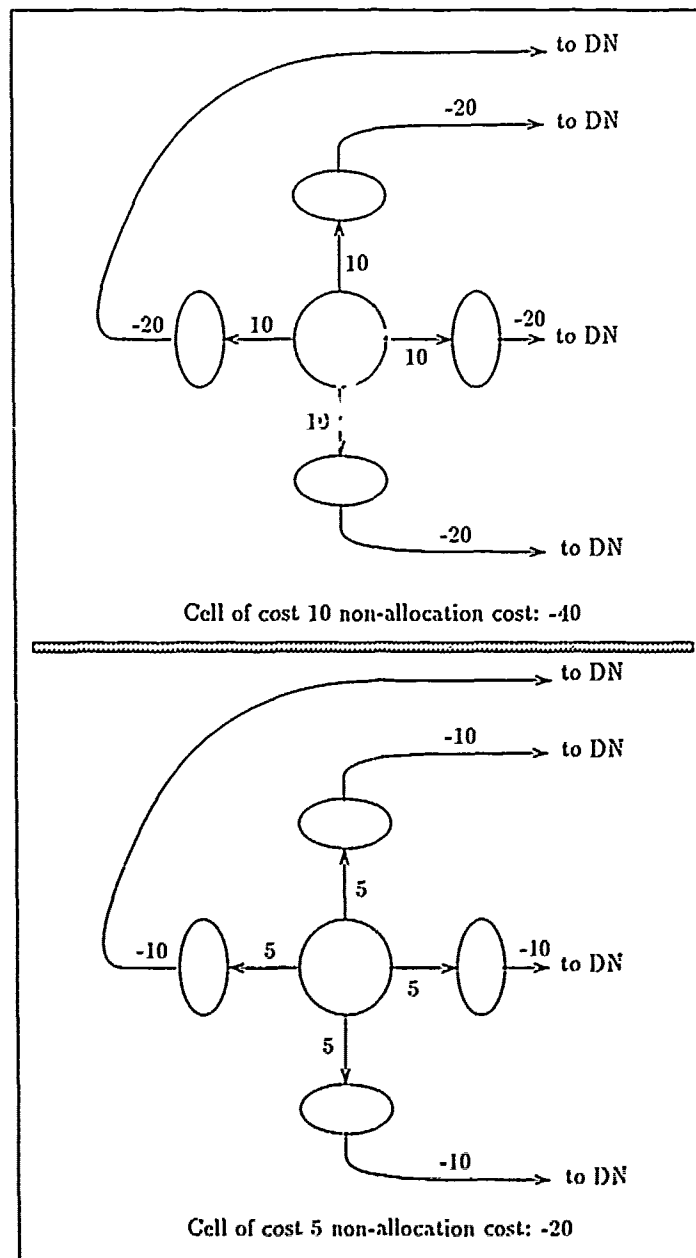


Figure 39. Non-Allocation Cost Assessment of the Cell-Border Network Version 2.



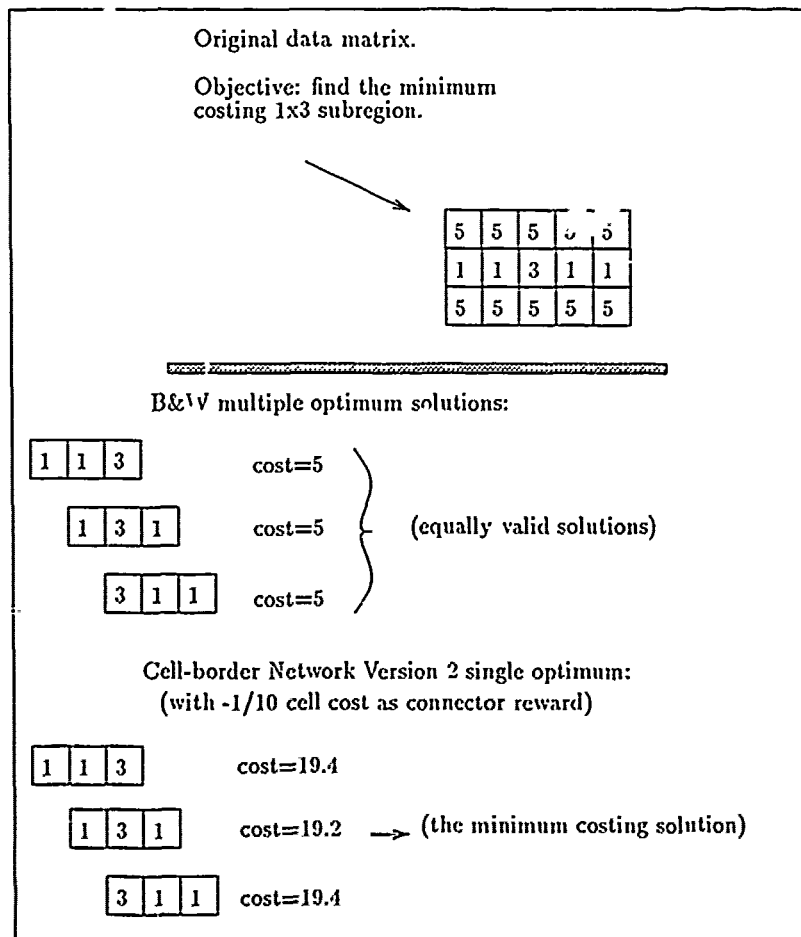


Figure 40. The Cell-Border Single Optimum vs. the B&W Multiple Optimum Solutions.

at the perimeter), is one way in which Version 2 of the Cell-Border network formulation *theoretically* exceeds the capabilities of the B&W models. The benefit is theoretical since actual, thorough, experimentation without tedious arc-setting is impossible without using a network package which can restrict flows on arcs to integer values. Never-the-less, the example shown in Figure 40 illustrates how the new allocation arc-cost assessments of the Version 2 Cell-Border Network would distinguish between the multiple optimum solutions of the B&W model to actually acquire, not only in terms of the area and compactness objectives, but also in terms of the distribution-of-cost objective, the single optimum solution.

1	2	3	4	5	6
7	(2)	(2)			12
13	(3)	(3)	(3)	(3)	18
19	(1)	(1)	(3)	(3)	24
25					30
31	32	33	34	35	36

$z=12$

Figure 41. CELLBORDER2B: Allocation Results From Run Number 7.

1	2	3	4	5	6
7	(2)	(2)			12
13	(3)	(3)	(3)	(3)	18
19	(1)	(1)	(3)	(3)	24
25					30
31	32	33	34	35	36

$z=-126$

Figure 42. CELLBORDER2B: Allocation Results From Run Number 8.

These two new cost assessment concepts were applied to problem CELLBORDER2B using both the Version-1 and Version-2 Cell-Border network formulations (see the partial input files in Appendix C showing the change in arc costs). The Version-1 allocation solution is shown in Figure 41 and the Version-2 allocation solution is shown in Figure 42. What can be derived from the allocation results of these

two runs is that nothing of significance has changed by adding artificial costs to the arcs in order to try to enforce total allocation of cells (thus the reason for including only their partial .log output files in Appendix C and not their .lis output files).

One other cost assessment scheme seemed worth attempting. Perhaps the cost of not allocating cells and their borders *should* be more than the cost of allocating them. Even though it is true that in the physical world, *not* allocating cells should not cost anything, it should be remembered that one of the original objectives is to maximize area; therefore, it would seem logical that the non-allocation artificial costs should penalize not allocating cells. The non-allocated cells, however, would still need to maintain their relative non-allocation worths: not allocating a cell of cost ten is better than not allocating a cell of cost one. The arc costs of the Wall-*B*-to-Node-*DN* arcs should, therefore, be the results of subtracting the cell cost from some pre-determined value.

Because all the cell costs in the example problem were single digits, it was decided that the non-allocation arc costs should be the difference between the number 100 and twice the original cell cost (see Figure 43 for an example); not allocating is penalized, and not allocating cheap cells is penalized more than not allocating expensive cells.

The allocation results of this new non-allocation cost assessment, based on the sum of a cell's flows to either Destination *DE* or to a connector node in Wall *C*, is shown in Figure 44. The partial input file and .log output file are located in Appendix C. Even though the solution has changed, no real allocation improvement has resulted. Since no further progress can be made on these relatively simple single subregion allocation problems, no attempts will be made to solve larger problems or multiple subregion problems.

#### 4.6 Determining the Most Compact Subregion

Since the subregion allocation problem is, in its most basic sense, a multi-criteria optimization problem (1:5-7), (minimizing the cost, maximizing the area, minimizing the perimeter, and the possible additional objective of centralizing the cost distribution) a method of running the network program without having to specify the shape and orientation of the desired subregion seemed in order. As the Cell-Border network formulation stood at this point, the user was required to input

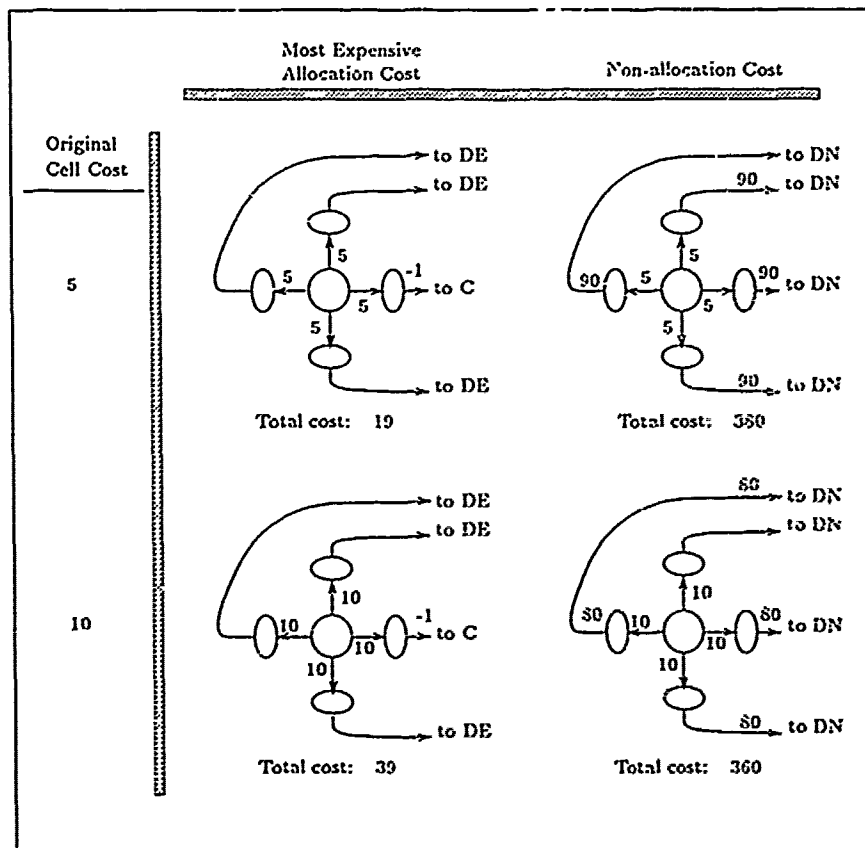


Figure 43. Final Non-Allocation Cost Assessment Example.

1	2	3	4	5	6
7	(3)	(3)			12
13	(3)	(3)	(3)	(3)	18
19	(1)	(1)	(2)	(2)	24
25					30
31	32	33	34	35	36

$z=3874$

Figure 44. CELLBORDER2B: Allocation Results From Run Number 9.

not only the desired subregion area, but also its height and width, restricting the program's use to finding only rectangular subregions which were oriented in a certain fashion (i.e.,  $2 \times 3$  versus  $3 \times 2$ ). The user could not simply find the minimum costing subregion of size 13, for instance.

Realizing that compactness is a property of the perimeter length of the subregion (4) as well as a property of the subregion internal border connections (in the Cell-Border network formulation), a means of determining the most compact subregion of a given area in terms only of the number of perimeter borders and internal connector borders was developed and implemented in the Cell-Border network. With the exception of those subregions which are able to be contained in a square (obviously the most compact subregion shape) no specific shape or orientation restrictions are applied to the subregion unless the user so chooses. Also, with no user input, the two Cell-Border network destination nodes used to tally internal row border connections and internal column border connections, Nodes *DIR* and *DIC*, are replaced with one node, *DI*, used to tally all the internal border connections with no restrictions placed on whether the connections are within rows or columns (thus, no orientation restrictions).

The method used to determine the most compact way to allocate the borders of a subregion is as follows (the specifics can be found in the program listing of the "cellborder" Pascal program in Appendix C):

- beginning with a  $1 \times 1$  artificial subregion, alternately increase the row value and the column value until the area is equal to or greater than the area of the desired subregion;
- if the artificial subregion area exceeds the desired subregion area, decrease the area of the artificial subregion by decreasing the row or column value (whichever is greater) by one (the cells making up the difference between the desired subregion area and the artificial subregion area are considered "left-over" cells for now;
- determine the number of perimeter borders and internally connected borders for this rectangular, possibly smaller-than-desired, embedded, artificial subregion;
- take the left-over cells which collectively can all fit along one side of the smaller artificial subregion (by virtue of the fact that the desired subregion area was exceeded in the first step) and determine the number of perimeter borders

- and internally connected borders which would be needed to add them to the artificial subregion rectangle;
- the value of the perimeter counter now holds the smallest perimeter the desired subregion area can fit into, and the value in the internal connector counter now holds the maximum number of connectors possible within the given perimeter.

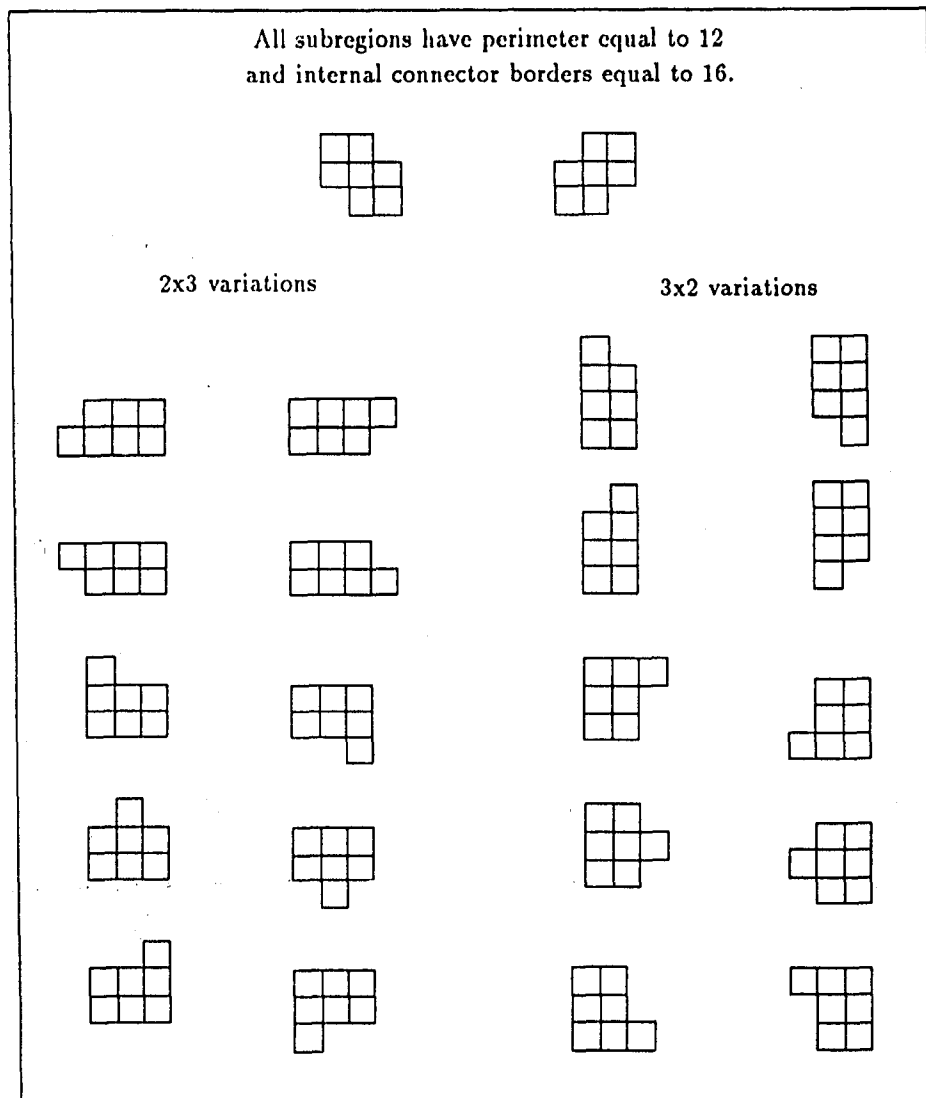


Figure 45. Size-7 Subregions Possible When Shape Constraints Are Not Specified.

Figure 45 illustrates the different ways cells can be allocated to a subregion of size 7 when maximum compactness is desired. Objectively, in terms of border connections and perimeter lengths, the compactness is identical (and maximum) for

all the allocation schemes shown in Figure 45; subjectively, some schemes may be considered "more" compact than others. The program is free to determine the minimum costing subregion of the desired area without having to meet fixed, unyielding requirements on the subregion's shape and orientation.

Because of this relaxing of the compactness constraint, a more multi-criteria approach to the solution can be made in a single model run than was previously possible in just one run. The only input required would be the area of the desired subregion; it is assumed that maximum compactness (in terms of perimeter length and internal border connections) is desired. No setting of shape or orientation is needed. In this way, several minimum-costing, most compact subregions of various shapes and orientations could be extracted from the input cost data and compared as far as compactness and cost are concerned. The many different orientations of a subregion with the same area, perimeter, and internal connections can represent the efficient frontier of solutions with the only variable remaining to be determined being that of shape and orientation.

1	2	3	4	5	6
7	(2.9)	(2.9)			12
13	(2.9)	(2.9)	(2.9)	(2.9)	18
19	(.6)	(.6)	(2.9)	(2.9)	24
25					30
31	32	33	34	35	36

$z=25.14$

Figure 46. CELLBORDER2B: Rounded-off Allocation Results From Run Number 10.

Problem CELLBORDER2B was run one more time using this new, unrestricted compactness formulation of the Cell-Border network in conjunction with Version-1 constraints (see the partial input listing in Appendix C). Again, the desired results were not obtained due to the inability to maintain integral flows from the border nodes. The allocation results are shown in Figure 46, and the partial output files are located in Appendix C.

#### 4.7 Conclusions

Any network formulation of the subregion allocation problem will have to derive integer flow solutions for there to be any improvements over the binary programming models in terms of faster solving speed, smaller problem complexity, and the ability to solve larger problems. If the split-flow problem could be overcome, then the Cell-Border network, which takes into account not only the data matrix cells but also the subregion perimeter and internal border connections, offers the fastest, most robust method of solving the subregion allocation problem. Larger problems could then be addressed with a much more rigorous exploration into the multi-criteria feasible region than is possible with the binary programming models.



## *V. Pixel Subregion Allocation: The Use of Subregion Allocation Concepts Towards Image Analysis*

### *5.1 Purpose*

Given that a person wants to analyze and model spatial-temporal information concerning a subregion of interest from satellite imagery, he must have the means with which to extract the spatial characteristics of that subregion from image to image over time in a consistent manner. It should be evident that when segmenting a satellite image into its natural subregions such as water, forest, farmland, and tundra, the shape and size of the subregions would not be known beforehand (assuming, of course, that spatial changes have taken place since the last segmentation). There will exist, however, an inherent spatial and spectral relationship between the pixels of any given subregion. A program which would allow a user to specify a spectral range of interest for a particular satellite image and then allow him to specify levels of compactness and configuity for any subregions whose pixels may lie within that spectral range would be most useful for those individuals attempting to perform spatial-temporal analysis. As an analogy, if one wanted to shoot someone, he would not aim at the arms or legs, but rather at the chest (unless, of course, only injuring the individual were the goal). A program which could find the "torso" of subregions and disregard any appendages if so desired would be useful to the spatial-temporal modellers. The subregion appendages might be indicative of small-scale, or short-term, trends such as reefs appearing around an island during low-tide.

This study illustrates the initial concepts of just such an approach for two different data files: 1) a single subregion contained within a 12-pixel by 12-pixel "pseudo-image" (i.e., the data is not from actual image data), and 2) a portion (20 × 20) of SPOT satellite imagery which has been pre-processed as mentioned in Chapter II. The General Algebraic Modeling System (GAMS) linear and non-linear programming package using the zero-one optimization option (ZOOM) within a VAX/VMS environment is used to solve the subregion allocation problems which make up the heart of this Pixel Subregion Allocation (PSA) study. Interaction between the user, the GAMS files, and the data files are accomplished via two pascal programs, SUBNETS.PAS and SOLUTION.PAS, and a VAX/VMS system program,

IMAGING.COM. Copies of the two new programs, IMAGING.COM and SOLUTION.PAS, as well as partial copies of the screen output of the model runs and GAMS input and output files are included in Appendix D.

## 5.2 The GAMS Single Subregion Allocation Model

Rather than focus on segmenting an existing image, this subregion allocation model will work from the premise that a completely empty "solution image" of the same size (same number of pixels) as the original image exists, and that at the conclusion of the model run, only those pixels of the solution image corresponding to the spectral and spatial requirements of the user will be "transferred."

Given an image of  $m$  rows and  $n$  columns, the binary variable  $x_{ij}$ ,  $i \in \{1, \dots, m\}$ ,  $j \in \{1, \dots, n\}$  will be used to represent the pixels.  $x_{ij}$  will equal 1 if the solution image pixel  $ij$  corresponds to a subregion pixel from the original image, and 0 if not.

Since the subregion allocation model is conceptually attempting to fill an empty solution image with the appropriate subregion pixels from the original image, the overall objective is to maximize the number of pixels transferred from the original image. This objective corresponds to maximizing area; thus, Equation 133 (1:5) is the overall objective.

$$\text{Max } Z = \sum_{i=1}^m \sum_{j=1}^n x_{ij} \quad (133)$$

To prevent the model from transferring every pixel from the original image into the solution image, objectives representing spatial requirements of the user will be applied. Only the candidate pixels (those lying within the grey value range of interest and denoted with a 1) will be subjected to the spatial constraints; therefore, every pixel of the working image whose cost is not equal to 1 will have its  $x_{ij}$  set to 0. Of the remaining candidate pixels, one of four possible spatial constraints based on the values of the  $x_{ij}$  variables of the pixel's neighbors can be applied.

It is necessary at this point to define what is meant by the terms "subregion" and "neighbor." Even though the underlying model for this PSA study is the Modified Single Subregion Allocation model (see Chapter II), a "single" subregion allocation, in the former sense of the word, is not necessarily the goal. In this image analysis study, a "single" subregion may well refer to several different non-contiguous

groupings of pixels within the satellite image. Each grouping meets the user's spatial requirements within itself, and all groupings fall within the user's spectral requirements (thus the "single" subregion concept is based on the fact that all groupings belong to the same spectral class). The term "neighbor" means the same for the pixels of the image as it did for the cells of a data matrix. First-order neighbors of a pixel are those pixels immediately above, below, to the left of, and to the right of a given pixel, and second-order neighbors are those pixels immediately diagonal (upper left and right, and lower left and right) of a given pixel.

To ensure that the spatial requirements of the user are met, the model need only check to ensure that each candidate pixel has the specified number and type of neighbor. The four compactness/contiguity user options and the GAMS constraints to meet those options follow (note how the actual value of the pixel itself is used to determine conformity):

option: Marginal compactness/contiguity requirements

- delete single pixel renegades:
- (pixels must have at least one 1st- or 2nd-order neighbor)

$$x_{(i-1,j-1)} + x_{(i-1,j)} + x_{(i-1,j+1)} + x_{(i,j+1)} + x_{(i+1,j+1)} \\ + x_{(i+1,j)} + x_{(i+1,j-1)} + x_{(i,j-1)} - x_{ij} \geq 0; \quad (134)$$

option: Low compactness/contiguity requirements

- pixels must have at least one 1st-order neighbor

$$x_{(i-1,j)} + x_{(i,j+1)} + x_{(i+1,j)} + x_{(i,j-1)} - x_{ij} \geq 0; \quad (135)$$

option: Medium compactness/contiguity requirements

- pixels must have at least two total neighbors

$$x_{(i-1,j-1)} + x_{(i-1,j)} + x_{(i-1,j+1)} + x_{(i,j+1)} + x_{(i+1,j+1)} \\ + x_{(i+1,j)} + x_{(i+1,j-1)} + x_{(i,j-1)} - 2(x_{ij}) \geq 0; \quad (136)$$

option: High compactness/contiguity requirements

- pixels must have at least two 1st-order neighbors

$$x_{(i-1,j)} + x_{(i,j+1)} + x_{(i+1,j)} + x_{(i,j-1)} - 2(x_{ij}) \geq 0 \quad (137)$$

where, for all equations above:  $i \in \{2, \dots, m-2\}, j \in \{2, \dots, n-2\}$

For any PSA problem, the binary programming problem which is solved by GAMS is made up of Equation 133 above, a constraint setting all non-candidate pixels'  $x_{ij}$  values to 0, and one of the above compactness/contiguity constraints.

### 5.3 The Programs and Files

The IMAGING.COM program serves three main functions: 1) executes the GAMS input file generator (the compiled Pascal program SUBNETS.EXE), 2) executes GAMS itself to solve the subregion allocation problem, and 3) executes the solution program (the compiled Pascal program SOLUTION.EXE). Initial concepts on the development of such programs came from Burke (4).

The imaging portion of the SUBNETS.PAS program serves three main functions: 1) queries the user for specific image data and for his spectral and spatial requirements, 2) converts the image file into a working file in which all pixels meeting the user's spectral requirements (candidate pixels) are given a cost (grey value) of 1, and those which do not are given an arbitrary large cost (88 in this case for on-screen visual purposes), and 3) creates the GAMS program file based on the user's spatial requirements. Initial concepts on the development of such programs came from Burke (4).

The SOLUTION.PAS program serves one main function: searches through the GAMS output file LISTING.DAT (which is a copy of the NSA.LIS generated by GAMS) to automatically determine the GAMS solution so that the solution can be stored and displayed in a suitable format for later use and user comparison.

The GAMS file, as has been mentioned earlier, solves an adaptation of a minimum cost, single subregion allocation problem (thus the need to set all candidate pixels' costs to 1 and all non-candidate pixels' costs to 88). There is not *one* GAMS file which is called by the IMAGING.COM program (as is the case with the Pascal programs); rather, a unique GAMS file must be written each time the PSA program is run so that new data sets and spatial requirements can be taken into account (recall that the spectral requirements of the user have already been incorporated by virtue of the manner in which the working data file was created). The GAMS program's effectiveness relies on the proper use of minimum cost subregion allocation concepts previously developed by Benabdallah and Wright (1) and further modified as discussed in Chapter III.

11	21	12	13	22	14	15	23	16	17	18	19	88	1	88	88	1	88	88	1	88	88	88	88
11	12	24	13	14	25	15	16	17	26	18	19	88	88	1	88	88	1	88	88	88	1	88	88
11	12	13	14	15	16	17	27	28	29	18	19	88	88	88	88	88	88	1	1	1	1	88	88
11	12	13	14	21	22	23	24	25	26	27	15	88	88	88	88	1	1	1	1	1	1	1	88
16	17	18	28	19	29	21	22	23	24	25	11	88	88	88	1	88	1	1	1	1	1	1	88
12	13	14	25	26	27	28	29	21	22	23	24	88	88	88	1	1	1	1	1	1	1	1	1
15	16	17	18	25	26	27	28	19	11	12	13	88	88	88	88	1	1	1	1	88	88	88	88
14	15	16	17	18	22	19	11	23	24	12	13	88	88	88	88	88	1	88	88	1	1	88	88
14	15	16	25	17	18	19	11	26	12	13	14	88	88	88	1	88	88	88	88	1	88	88	88
15	16	17	18	19	11	12	13	14	15	16	17	88	88	88	88	88	88	88	88	88	88	88	88
18	19	11	22	12	13	14	23	15	16	17	18	88	88	88	1	88	88	88	1	88	88	88	88
19	11	12	13	14	15	16	17	18	19	11	12	88	88	88	88	88	88	88	88	88	88	88	88

Figure 47. The Pseudo-Image Input Data and (Frame-less) Working Data.

#### 5.4 The Example Runs

Using the pseudo-image data (Figure 47) and the programs listed above, one model run for each of the compactness/contiguity options was performed wherein pixels whose grey values lay between 20 and 30 (inclusive) were allocated to a subregion (subject to also meeting the spatial requirements). Only partial input and output files and screen output for the "Low" and "High" compactness/continuity options are included in Appendix D. Solution images for the "Low" and the "High" compactness/continuity options are shown in Figure 48. The solutions are not a debatable issue; suffice it to say that the subregions were in fact brought under more and more stringent contiguity and compactness constraints as higher levels of the spatial options were chosen; all the while, the objective to maximize the subregion's area within these spatial constraints was successfully met.

#### 5.5 Subregion Allocation of Real Satellite Imagery

Figure 49 shows the pixel grey values of the real satellite imagery used in the subregion allocation experiment. The  $20 \times 20$  matrix of pixels represents approximately a  $4\text{km} \times 4\text{km}$  area of the SPOT satellite image of the Washington, D.C. region. More information on the image can be found in Chapter II. Using this data and the programs listed above, one model run for the "High" compactness/contiguity option was performed wherein pixels whose grey values lay between 200 and 255 (inclusive) were allocated to a subregion (subject to also meeting the spatial requirements).

Figure 48. Pseudo-Image Solutions for Low and High Compactness Options.

Figure 49. The Real Image Input.

```

88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 1 1 1 1 1
88 88 88 88 88 88 88 88 88 88 88 88 88 1 88 1 1 1 1 1
88 88 88 88 88 88 1 88 88 88 88 1 1 1 1 1 1 1 1 1
88 88 88 88 88 88 88 88 88 88 88 88 1 88 1 1 1 1 1
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 1
1 1 88 88 1 1 88 88 88 88 88 88 88 1 1 1 88 88 1
1 1 1 88 1 1 88 88 88 88 88 88 88 1 1 1 1 88 88 88
1 1 1 1 1 1 1 88 88 88 88 88 88 1 1 1 1 88 88 88
1 1 1 1 1 1 1 88 88 88 88 88 88 1 1 88 88 88 88
1 1 1 1 1 88 88 88 1 1 88 88 88 88 88 88 88 88 88
1 88 88 88 88 88 88 88 88 1 1 1 1 1 88 88 88 88 88
1 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 1 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 1 1 88 88 88 88 88 88 88 88 88 88 88 88

```

Figure 50. The Real Image (Frame-less) Working Data.

The working data file is shown in Figure 50. The spectral range was chosen solely to illustrate the point that any range in any kind of imagery may indicate the presence of a particular class of terrain (the specific range of which would be known to the user). This example is in no way implying that there is some specific terrain class within the spectral range in this imagery. Appendix D contains a partial listing of the screen output and the GAMS output file.

The solution image for the "High" compactness/continuity option is shown in Figure 51. Had the spectral range actually been the range for a specific terrain class, then it would be evident that three such areas are included in the image.

### 5.6 Conclusions

The only "problem" (long execution time) occurred with a different input data file when attempting to find "subregions" (the word is used loosely) when there was little natural compactness already inherent in the image (trying to find the patterns of a random number generator would be a comparable feat). This is not actually a problem of the model, since the creation of major subregions from images where none exist is not the goal of the program. Continuing work needs to be done in the areas of 1) finding a more suitable programming model with which to apply the subregion allocation constraints (perhaps a network formulation), and 2) allowing for compactness constraints of the allocated areas to be determined more on a subregion-wide basis than solely on a pixel-by-pixel basis in order to decrease processing time.

The example applications shown in this chapter are in no way meant to be a complete exploration of the feasible solutions to a subregion allocation problem of satellite imagery. Referencing discussions that can be found in Chapters III and IV, a full treatment of the multiple objectives of subregion allocation can only be accomplished through either combining all the objectives into one overall objective statement, or (as was the case with the subregion allocation models previously used) exploring the efficient frontier of solutions that are possible after restating certain objectives as constraints and setting bounds on their values.

As for the actual use of the above PSA model, it could serve as the first step in such an efficient frontier search. The subregion determined through the PSA model (objective: maximize area) could be the baseline subregion(s) for a full implementation of the subregion allocation models (objective: minimize cost). An



\*\*\*\*\*

# SOLUTION IMAGE:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
2	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	1	1	1	1	1
3	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	1	1	1	1	1
4	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	1	1	1	1	1
5	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	1	1	1	1	1
6	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
7	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
8	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
9	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
10	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
11	1	1	88	88	1	1	88	88	88	88	88	88	88	88	1	1	1	88	88	88
12	1	1	1	88	1	1	88	88	88	88	88	88	88	1	1	1	1	88	88	88
13	1	1	1	1	1	1	1	88	88	88	88	88	88	1	1	1	1	88	88	88
14	1	1	1	1	1	1	1	88	88	88	88	88	88	88	1	1	88	88	88	88
15	1	1	1	1	1	1	1	88	88	88	88	88	88	88	1	1	88	88	88	88
16	1	1	1	1	1	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
17	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
18	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
19	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
20	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88

NOTE: Pixels denoted with a 1 represent actual subregion pixels as dictated by your spectral and spatial requirements; those denoted with 88 are non-subregion pixels.

The subregion pixels comprise a total area of: 70

\*\*\*\*\*

Figure 51. The Real Image Solution for the High Compactness/Contiguity Option.

algorithm could be developed which assigns artificial values to the cells based on their spatial relationships to the PSA-extracted subregion in a manner similar to the way in which cells were assigned artificial values based on their spectral relationships to the desired grey-value range. The spatially-based artificial cost assignments would, of course, be more complex than the simpler spectrally-based cost assignments used in the PSA model. The "1's", previously used to represent cells meeting the spectral requirements of the user, could now represent those cells comprising a spatially contiguous and compact subregion. Additional values (2, 3, ...) could be assigned to the remaining pixels based on their spatial relationships to the subregion of 1's. In this way, the subregion allocation programs (based on the overall objective of minimizing cost) could be run on this data of cells whose costs are 1, 2, 3, ..., and an efficient frontier of solutions could be obtained by varying the desired size, perimeter, and shape/orientation of the subregions.

## VI. Conclusions and Recommendations

### 6.1 Subregion Allocation Through Binary Programming

It is not possible to take into account all objectives of the subregion allocation problem (minimizing cost, maximizing area, and maximizing compactness (1:5-6)) in a true multiobjective treatment without using the constraints and variables in the B&W formulation implemented via some means such as the Burke implementation. Without taking into account each border of every cell, it was determined that the best formulation for the modified subregion allocation problems would still require more binary variables than simply those representing the cells themselves (binary variables for rows and columns as well as additional variables and constants representing desired subregion sizes). Even though the example problems were solved quicker this way, it was subsequently determined that, based on the tremendous increase in branch-and-bound iterations required to reach the solution, any savings in computation time would soon be overcome by further branch-and-bound iteration increases as larger problems were encountered. These experiments in variable re-definitions and constraint reductions of the modified models indicate the overall conclusion just mentioned, that the constraints and variables found in the B&W model are the most appropriate to use in order to fully treat the multiobjective basis of the subregion allocation problem.

Through the experiments, it was learned that desired subregion allocation solutions for specific applications which may not require a full treatment of all the objectives can be achieved by using modified versions of the B&W subregion allocation formulation (the pixels of satellite imagery of a specific terrain area already possess some measure of contiguity in an area-wide sense, so cell-by-cell compactness and contiguity constraints in lieu of subregion perimeter constraints may be sufficient). In fact, for some applications, such as the allocation of imagery pixels into terrain classes, a modified version must be used because:

- the objective of the problem is not to minimize cost based on fixed spatial requirements but, rather, to maximize the allocation based on a fixed cost (spectral) range,

- the desired size and shape of the subregion is not known (precisely) before the allocation has occurred, and
- several areas may make up one "subregion" (reference the re-definition of the word in Chapter V), a situation which lies somewhere between a B&W single subregion problem (all areas belong to the same spectral class) and a multiple subregion problem (each separate area is not spatially contiguous to any other area of the subregion).

## 6.2 Subregion Allocation Through Networks

Almost any problem that can be formulated as a network can probably be formulated as any of a hundred different networks because each piece of data and each part of the problem statement could conceptually be modelled as nodes, arcs, arc costs, flows, source supplies, destination demands, or side constraints in any of several different networks such as min-cost, shortest-path, max-flow, or matching. For the subregion allocation problem, however, one certainty exists: because of the determination that cell borders do need to be taken into account to fully treat the multiobjective basis of the problem, any network formulation of the problem must also take cell borders into account. For this reason, the Cell-border network-with-side-constraints formulation is the only network developed in this research that conceptually possesses the ability to solve the subregion allocation problem in its full, multiobjective sense.

Because network formulations of problems are much less complex than their similar-sized LP and BP counterparts (7), a network formulation of the subregion allocation problem should be a much quicker and more efficient way to solve the problem than the LP or BP formulation.

Generally speaking, network problems are much less complex than their comparable linear and binary programming problems, comparing on the order of complexity of  $n \times \binom{m+n}{m}$  (where  $n$  is the number of nodes and side constraints and  $m$  is the number of arcs) for networks versus  $mn \times \binom{m+n}{m}$  (where  $m$  and  $n$  are the number of equations and variables) for binary programming (7). Because the node-arc incidence matrix of a network is the comparable construction to the equation-variable matrix of linear and binary programming, the side constraints should also count with

the total tally of nodes since they do add to the number of network equations in the node-arc incidence matrix.

For the ideal (integer flows on all arcs which have flow) Cell-Border formulation of the subregion allocation problem, the only side constraints needed would be those stating that two adjacent cell borders must pass their one unit of flow on to their connector node (set one) and those requiring that each border of an allocated cell contribute the same amount (set two). The number of nodes, arcs, and side constraints for any SSA Cell-Border network could be determined from the following functions (where  $n$  is the number of cells in a rectangular matrix,  $row$  is the number of matrix rows, and  $col$  is the number of matrix columns):

$$\#nodes = (\# \text{ of cells}) + (\# \text{ of cell borders}) + (\# \text{ of connector nodes}) + (\text{source node}) + (\text{destination nodes})$$

$$\begin{aligned}\# \text{ nodes} &= n + 4n + (2n - row - col) + 1 + 3 \\ &= 7n + 4 - row - col\end{aligned}$$

$$\#arcs = (\text{source arcs}) + (\text{M to B arcs}) + (\text{B to C arcs}) + (\text{B to DE arcs}) + (\text{C to DIR/C arcs})$$

$$\begin{aligned}\# \text{ arcs} &= n + 4n + (4n - 2row - 2col) + 4n + (2n - row - col) \\ &= 15n - 3row - 3col\end{aligned}$$

$$\#side \text{ constraints} = \# \text{ of connector nodes} + 3 \times (\# \text{ cells})$$

$$\begin{aligned}\# \text{ set-one and set-two side constraints} &= (2n - row - col) + 3n \\ &= 5n - row - col\end{aligned}$$

The Burke implementation of the SSA problem would also require a certain number of variables and equations based on the following functions (where  $n$  is the number of cells in a rectangular matrix):

$$\#variables = (\# \text{ of cells}) + 2(\# \text{ of cell borders}) + (\text{the objective})$$

$$\# \text{ variables} = n + (2 \times (4n)) = 9n + 1$$

$$\#equations = (\text{area eq} + \text{perimeter length eq}) + (\text{total } P \text{ and } N \text{ eqs}) + (\text{cell allocation eqs}) + (\text{the objective})$$

$$\# \text{ equations} = 2 + 8n + n + 1 = 9n + 3$$

Using the functions and the complexity relationships described above, a comparison of the complexity for the LP versus the network SSA problem formulations can be made for any square cost matrix of size  $row$  and  $col$  where  $row \times col = n$ . The following complexity value calculations only take into account the more significant

contributors to the relative complexity values (i.e., the smaller numbers have been left out):

Complexity of the BP formulation:

$$(9n + 3)(9n + 1) \times \begin{pmatrix} (18n + 4) \\ (9n + 3) \end{pmatrix}$$

which can be re-written as being roughly equivalent to:

$$(81n^2 + 36n) \times \begin{pmatrix} (18n) \\ (9n) \end{pmatrix}$$

Complexity of the Network formulation:

$$((7n) + 4 - row - col) + ((5n) - row - col) \times \begin{pmatrix} (27n) + 4 - (5row) - (5col) \\ (12n) + 4 - (2row) - (2col) \end{pmatrix}$$

which can be re-written as being roughly equivalent to:

$$(12n) \times \begin{pmatrix} (27n) \\ (12n) \end{pmatrix}$$

Although the BP formulation seems to be on the order of the square of the comparable Cell-Border network formulation, it is the combinatorial portion of the formula which carries the most weight (7). In this light, the network is more complex than the BP counterpart. It is assumed, however, that as problem size increased, and the problem became one of multiple subregion allocation ( $K$  subregions) instead of single subregion allocation, the combinatorial portion of the BP complexity function would approach and surpass that of the network function, thus rendering the Cell-Border network formulation complexity value at least as small as the square root of the complexity value of the BP formulation. It is possible that the Cell-Border network would deal with multiple subregions in a manner similar to the way in which the Area-Wall formulation dealt with multiple subregions (same network, additional destinations). This would result in the following "on-the-order-of" (most significant contributor) changes to the functions and complexity values.

For the Cell-Border network, the only significant change would be in the number of arcs:

$$\begin{aligned} \#arcs = & (\text{source arcs}) + (M \text{ to } B \text{ arcs}) + (B \text{ to } C \text{ arcs}) + (B \text{ to } (K)(DE) \text{ arcs}) \\ & + (C \text{ to } (K)(DIR/C) \text{ arcs}) \end{aligned}$$

$$\begin{aligned}\# \text{ arcs} &= n + 4n + (4n) - 2\text{row} - 2\text{col} + 4Kn + (K)(2n - \text{row} - \text{col}) \\ &\simeq 9n + 6Kn\end{aligned}$$

The BP formulation would increase in both the number of variables and equations:

$$\begin{aligned}\# \text{ variables} &= (\# \text{ of } (K)(\text{cells}) ) + 2(\# \text{ of } (K)(\text{cell borders}) ) + (\text{the objective}) \\ \# \text{ variables} &\simeq 9Kn\end{aligned}$$

$$\# \text{ equations} = (K)(\text{area} + \text{perimeter eqs}) + (\text{total } (K)(P \text{ and } N \text{ eqs}) ) + (\text{cell allocation eqs}) + (\text{the objective})$$

$$\# \text{ equations} \simeq 8Kn + n$$

As an example, the complexity values for a MSA problem where  $K = 5$  and  $n = 100$  would be as follows:

Complexity of the BP formulation:

$$(9Kn)(8Kn + n) \times \binom{(9Kn + 8Kn + n)}{(9Kn)}$$

which approximately equals:

$$(4500)(4100) \times \binom{8600}{4500}$$

Complexity of the Cell-Border Network formulation:

$$(12n) \times \binom{(12n) + (9n + 6Kn)}{(12n)}$$

which approximately equals:

$$(1200) \times \binom{5100}{1200}$$

The Cell-Border network complexity value, theoretically, would be many magnitudes less than the BP formulation. Even if several new sets of side constraints would be needed, it is apparent that many hundred side constraints for the example problem (many sets on the order of  $n$ , in general) could be added without the network formulation losing its advantage in complexity.

### 6.3 *The Multiple Objectives of Subregion Allocation*

The B&W model will find the minimum costing subregion of a specific rectangular shape, but it will also allow the user to specify only a subregion perimeter length, thus acquiring the minimum costing subregion of area  $M$  (and the most compact if the user specified the perimeter length  $P$ ) without the need to specify an exact-subregion shape and orientation. If the user wants the most compact subregion he can get, however, he will need to determine what that subregion would look like and then calculate the perimeter length. Conceptually, the Cell-Border model will do the same, but without the user having to figure out a perimeter length. Built into the program are the calculations necessary to determine the most compact subregion of area  $M$  (the only user input required) based on perimeter length and internal border connections; the model can find the minimum costing, most compact subregion of area  $M$  without needing the user to determine the perimeter first. Maximizing compactness can be dealt with simultaneously while minimizing cost.

The Cell-Border network can also, conceptually, distinguish between subregions based on cost distribution. Although not built in to the Cell-Border program, anyone can alter the way in which the program assesses arc costs, and thus include "centralizing cost distribution" as an additional objective of subregion allocation.

### 6.4 *The Spatial Analysis of Imagery*

No network was implemented for the task of spatial analysis of imagery; however, the BP formulations of the subregion allocation problem were experimented with and modified so as to be able to apply the basic subregion allocation concepts towards that goal. Further work should be done to enhance the imaging program to allow for a more subregion-wide treatment of compactness rather than the cell-by-cell treatment currently employed. To note is the fact that the term "subregion" used in the imaging application very probably refers to more than one area, or clustering, of pixels of the image.

As for the actual use of the PSA model, it could serve as the first step in an efficient frontier search. The subregion determined through the PSA model (objective: maximize area) could be the baseline subregion(s) for a full implementation of the subregion allocation models (objective: minimize cost). Details concerning the development of an algorithm which could assign artificial values to cells based on



their spatial relationships to the PSA-extracted subregion were discussed in Chapter V. It is possible that, following the subregion extraction based on the spectral range and cell-based spatial requirements (the PSA model), a subregion allocation, min-cost approach could be applied to the cells to acquire a full, multi-objective efficient frontier of solutions based on varying the area and perimeter lengths of the subregions.

### 6.5 *Final Recommendations and Conclusions*

The arc-setting requirement of a branch-and-bound solution approach to "fix" split flows (partial cell allocations to subregions) needs to be changed. It is suggested that, rather than adding a side constraint requiring a fixed amount of flow on an arc, the better course of action would be to change the network structure itself. The flow from the source (or previous wall of nodes) would be decreased by the desired flow on the arc to be set, and the head-node of the arc that would have been set would actually supply the desired flow into the remainder of the network. This would be a simple enough task if the arcs to be set originated at the source; however, more thought would be required in order to implement such a structural change into the Pascal programs to allow for any arc in the network to be "set" in this way. Advantages would be 1) a reduction in the number of side constraints (thus shorter solution times), and 2) the structural changes would reduce the size of the network (and thus the complexity) more and more for each arc that is set.

The most fundamental problem in the application of the network formulation was an inability to maintain integral flows on arcs which had flow. The Area Wall program had to be run again and again to branch-and-bound towards a solution. Even though the Cell-Border network complexity value would be much less than that of the comparable BP, and many, many runs could be made without losing that particular advantage, the real-time task of re-running the program over and over would be exceedingly tiresome. A network-with-side-constraints package that could maintain integral flows seems almost a necessity.

Any network formulation of the subregion allocation problem will have to derive integer flow solutions for there to be any improvements over the binary programming models in terms of faster solving speed, smaller problem complexity, and larger problem size. If the split-flow problem could be overcome, then the Cell-Border network, which takes into account not only the data matrix cells but also the subregion

perimeter and internal border connections, offers the fastest, most robust method of solving the subregion allocation problem. Larger problems could then be addressed with a much more rigorous exploration into the multi-criteria feasible region (including compactness and cost distribution) than is possible now with the binary programming models.

## Appendix A. *Instructions for Running the Computer Programs*

### A.1 *BP Programs*

#### A.1.1 *Running a GAMS Program.*

1. Create the GAMS input file using whatever editor on whatever system you choose.
2. Copy or transfer your GAMS input file to the VAX/VMS system under the name [your filename].gms.
3. Login to the VAX/VMS operating system.
4. Issue the command `getgams`  
(only needed once per login).
5. Issue the command `gams [your filename]`.
6. The GAMS solver will process the input file.  
(The window/screen you are using will be locked up while the GAMS solver solves your problem (unless the problem is submitted via a batch job, of course.) )
7. Whether a solution is reached or not, the output will be listed in the file [your filename].lis.

#### A.1.2 *The Burke Implementation of the B&W Models.*

1. Login to the VAX/VMS operating system.
2. Issue the command `getgams`  
(only needed once per login).
3. Create your input matrix (cost data) file under any name you wish.  
(You will be prompted for the file name later.)
4. Issue the command `@msa`, then
  - Enter in your input data file name, when prompted for cost matrix;
  - Enter in any name you wish the GAMS file to use ("`[filename].gms`" by convention), when prompted for GAMS file;
  - Pick the solution model type you wish to use, when prompted;
  - Enter in the number of input data rows and columns, when prompted;
  - Enter in the desired number of subregions, desired perimeter length for each, and height and width (if required) as you are prompted for the information;

- The GAMS solver will process the input file.  
(The window/screen you are using will be locked up while the GAMS solver solves your problem.)
5. Whether a solution is reached or not, the output will be listed in the file [filename].lis.

### A.1.3 The Imaging Programs.

1. Login to the VAX/VMS operating system.
2. Create your input matrix (image grey-scale data) file under the name "image.dat" (it should be a text file; i.e., if you "type" or "more" it, it should be readable.)
3. Issue the command @imaging.  
(The GAMS input file is automatically created under the name "nsa.dat" and will later be deleted.) Then,
  - Enter in the number of input data rows and columns, when prompted;
  - Enter in 1 (for Image Subregion Determination), when prompted for your program option;
  - Enter in the minimum and maximum grey values of the spectral range you are interested in, when prompted;
  - Enter in the number corresponding to the compactness/contiguity option you wish, when prompted;  
(The other compactness/contiguity options can be selected by re-running the program from the start.)
  - The GAMS solver will process the input file.  
(The window/screen you are using will be locked up while the GAMS solver solves your problem.)
4. The allocation solution will be visually displayed on the screen. A copy of the allocation solution will be stored in the file "compactim.dat". The actual GAMS output (which includes a copy of the input file) will be in the file "listing.dat". Temporary data files used by the program and all but the most current version of the "listing.dat" file are deleted. None of the "compactim.dat" files are deleted, however, so that if several runs are needed, all allocation solutions will be available.

## *A.2 Network Programs*

### *A.2.1 Running a SAS Program*

1. Create the SAS input file using whatever editor on whatever system you choose.
2. Copy or transfer your SAS input file to the VAX/VMS system under the name [your filename].sas.
3. Login to the VAX/VMS operating system.
4. Issue the command runsas [filename].
5. Press [RETURN], when prompted for the VMS data file.
6. Press [RETURN] when prompted for the queue you wish to submit your program to if you want to submit it to the short SAS queue (time limit: 2 CPU minutes). If you want to submit your program to the long SAS queue (time limit: 60 CPU minutes), enter L, and press [RETURN].
7. The SAS solver will process the input file.  
(The window/screen you are using will NOT be locked up since the program has been submitted like a batch job.)
8. You will be prompted on screen when the SAS job has completed. Whether a solution is reached or not, the output will be listed in the files [your filename].log and [your filename].l3.

### *A.2.2 Running the Area-Wall Network Program*

1. Login to the VAX/VMS operating system.
2. Create your input matrix (cost data) file under the name "image.dat".
3. Issue the command run subnets, then
  - Enter in the number of input data rows and columns, when prompted;
  - Enter in 2 (for Min-Cost Subregion Determination), when prompted for your program option;
  - Enter in the number of subregions you wish to determine from your input data, when prompted;
  - Enter in the desired area for each of your subregions, in turn, when you are prompted.
4. Issue the command run areawall, then
  - Enter in the number corresponding to the compactness option you want, when prompted;

- Enter in the number corresponding to your present situation option
5. Type `runsas nsa.dat` when the system prompt appears.
  6. Follow the numbered instructions on the screen as needed.  
(They cover such topics as re-running the program, setting arcs, and viewing the solution.)
  7. The solution to the SAS program will be contained in the files "nsa.log" and "nsa.lis".

### *A.2.3 Running the Cell-Border Network Program.*

1. Login to the VAX/VMS operating system.
2. Create your input matrix (cost data) file under the name "image.dat".
3. Issue the command `run subnets`, then
  - Enter in the number of input data rows and columns, when prompted;
  - Enter in 2 (for Min-Cost Subregion Determination), when prompted for your program option;
  - Enter in 1, when prompted for the number of subregions you wish to determine;  
(Multiple subregions are not allowed in the Cell-Border formulation.)
  - Enter in the desired area for your subregion, when prompted.
4. Issue the command `run cellborder`, then
  - Read the directions for option selecting;
  - Enter in the number(s) corresponding to the program option(s) you want, one at a time, on separate lines, when prompted.  
(Note restrictions.)
5. Type `runsas nsa.dat` when the system prompt appears.
6. Follow the numbered instructions on the screen as needed.  
(They cover such topics as re-running the program, and viewing the solution.)
7. The solution to the SAS program will be contained in the files "nsa.log" and "nsa.lis".

## Appendix B. Files for the B&W and Modified Subregion Allocation Problems

### B.1 Partial Output File for Problem SSA1

GAMS 2.20 VAX VMS 12-OCT-1991 13:49  
GENERAL ALGEBRAIC MODELING SYSTEM  
COMPI LATION

```
2 SETS
3   I / 1*16/
4   K / 1*1/
5 ALIAS(I,J);
6 SETS
7   ADJ(I,J) ADJACENCY MATRIX
8       /1.(13,4,2,5)
9       2.(14,1,3,6)
10      3.(15,2,4,7)
11      4.(16,3,1,8)
12      5.(1,8,6,9)
13      6.(2,5,7,10)
14      7.(3,6,8,11)
15      8.(4,7,5,12)
16      9.(5,12,10,13)
17      10.(6,9,11,14)
18      11.(7,10,12,15)
19      12.(8,11,9,16)
20      13.(9,16,14,1)
21      14.(10,13,15,2)
22      15.(11,14,16,3)
23      16.(12,15,13,4)/
24
25 PER(I,J) PERIMETER MATRIX
26     /1.(13,4)
27     2.(14)
28     3.(15)
29     4.(16,1)
30     5.(8)
31     8.(5)
32     9.(12)
33     12.(9)
34     13.(16,1)
35     14.(2)
36     15.(3)
37     16.(13,4)/
```

```

38
39     EXC(I,J);
40
41     EXC(I,J) = ADJ(I,J) - PER(I,J);
42
43     PARAMETERS
44         M(K) CELLS
45         / 1 2/
46
47         L(K) BORDERS
48         / 1 6/
49
50         C(I) COST MATRIX
51         / 1 6, 2 5, 3 3, 4 4
52           5 1, 6 3, 7 2, 8 2
53           9 5, 10 4, 11 1, 12 3
54          13 2, 14 2, 15 3, 16 1/;
55
56     VARIABLES
57         Z TOTAL COST OF ACQUIRED CELLS
58         X(I,K)
59         P(I,J,K)
60         N(I,J,K)
61
62     BINARY VARIABLES X,P,N;
63
64     EQUATIONS
65         OBJ          DEFINE OBJECTIVE FUNCTION
66         C1(K)        CONSTRAINT FOR NUMBER OF ACQUIRED CELLS
67         C2(I,J,K)    CONSTRAINT FOR ALL BORDERS
68         C3(K)        CONSTRAINT FOR BORDER LENGTH
69         C4(I,J,K)    CONSTRAINT FOR P&N MUTUAL EXCLUSION
70         C5(I,J,K)    CONSTRAINT FOR PERIMETER BORDERS
71         C6(I)        CONSTRAINT FOR SUBREGIONS;
72
73     OBJ..
74         Z =E= SUM((I,K), C(I)*X(I,K));
75
76     C1(K)..
77         SUM(I,X(I,K)) =E= M(K);
78
79     C2(I,J,K) $ADJ(I,J)..
80         X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
81
82     C3(K)..
83         SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =E= 2*L(K);
84
85     C4(I,J,K) $EXC(I,J)..
86         P(I,J,K) + N(I,J,K) =L= 1;
87
88     C5(I,J,K) $PER(I,J)..

```



```

89:   X(I,K) - P(I,J,K) =E= 0;
90:
91:   C6(I) = .
92:   SUM(K, X(I,K)) =L= 1;
93:
94:   MODEL SUBREGION /ALL/;
95:
96:   OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 20000;
97:
98:   SOLVE SUBREGION USING MIP MINIMIZING Z;
99:
100:  OPTION MIP=ZOOM;
101:
102:  DISPLAY X,L,P,L,N,L;
103:

```

#### MODEL STATISTICS

BLOCKS OF EQUATIONS	7	SINGLE EQUATIONS	147
BLOCKS OF VARIABLES	4	SINGLE VARIABLES	145
NON-ZERO ELEMENTS	561	DISCRETE VARIABLES	144

GENERATION TIME = 1.100 SECONDS

EXECUTION TIME = 1.730 SECONDS

#### S O L V E S U M M A R Y

MODEL	SUBREGION	OBJECTIVE	Z
TYPE	MIP	DIRECTION	MINIMIZE
SOLVER	ZOOM	FROM LINE	98

```

**** SOLVER STATUS      1 NORMAL COMPLETION
**** MODEL STATUS       1 OPTIMAL
**** OBJECTIVE VALUE          3.0000

```

RESOURCE USAGE, LIMIT	9.790	1000000.000
ITERATION COUNT, LIMIT	592	1000000

Z O O M / X M P --- Version 2.1 Oct 1988

Courtesy of Dr Roy E. Marsten,  
 Department of Management Information Systems,  
 University of Arizona,  
 Tucson Arizona 85721, U.S.A.

No options file found - using defaults.

Work space needed (estimate) -- 56906 words.

Work space available -- 56906 words.

The branch and bound tree contained 59 nodes (max. 20000 nodes).

Iterations: Initial LP	151,	Time: 1.55000
Heuristic	5,	0.880001
Branch and bound	435,	6.32000
Final LP	1,	9.000015E-02

\*\*\*\* REPORT SUMMARY :  
0 NONOPT  
0 INFEASIBLE  
0 UNBOUNDED

----- 102 VARIABLE X.L

1

7	1.000
11	1.000

## B.2 Input File for Problem MSA1

GAMS 2.20 VAX VMS

12-OCT-1991 14:05

GENERAL ALGEBRAIC MODELING SYSTEM  
COMPI LATION

```
2 SETS
3   I / 1*36/
4   K / 1*2/
5 ALIAS(I,J);
6 SETS
7   ADJ(I,J) ADJACENCY MATRIX
8       /1.(31,6,2,7)
9       2.(32,1,3,8)
10      3.(33,2,4,9)
11      4.(34,3,5,10)
12      5.(35,4,6,11)
13      6.(36,5,1,12)
14      7.(1,12,8,13)
15      8.(2,7,9,14)
16      9.(3,8,10,15)
17      10.(4,9,11,16)
18      11.(5,10,12,17)
19      12.(6,11,7,18)
20      13.(7,18,14,19)
21      14.(8,13,15,20)
22      15.(9,14,16,21)
23      16.(10,15,17,22)
24      17.(11,16,18,23)
25      18.(12,17,13,24)
26      19.(13,24,20,25)
27      20.(14,19,21,26)
28      21.(15,20,22,27)
29      22.(16,21,23,28)
30      23.(17,22,24,29)
31      24.(18,23,19,30)
32      25.(19,30,26,31)
33      26.(20,25,27,32)
34      27.(21,26,28,33)
35      28.(22,27,29,34)
36      29.(23,28,30,35)
37      30.(24,29,25,36)
38      31.(25,36,32,1)
39      32.(26,31,33,2)
40      33.(27,32,34,3)
41      34.(28,33,35,4)
42      35.(29,34,36,5)
43      36.(30,35,31,6)/
44
45 PER(I,J) PERIMETER MATRIX
```

```

46      /1.(31,6)
47      2.(32)
48      3.(33)
49      4.(34)
50      5.(35)
51      6.(36,1)
52      7.(12)
53      12.(7)
54      13.(18)
55      18.(13)
56      19.(24)
57      24.(19)
58      25.(30)
59      30.(25)
60      31.(36,1)
61      32.(2)
62      33.(3)
63      34.(4)
64      35.(5)
65      36.(31,6)/
66
67      EXC(I,J);
68
69      EXC(I,J) = ADJ(I,J) - PER(I,J);
70
71      PARAMETERS
72      M(K) CELLS
73      / 1 9, 2 4/
74
75      L(K) BORDERS
76      / 1 12, 2 8/
77
78      C(I) COST MATRIX
79      / 1 6, 2 5, 3 3, 4 4, 5 3, 6 4
80      7 2, 8 5, 9 1, 10 8, 11 5, 12 3
81      13 7, 14 7, 15 3, 16 4, 17 9, 18 2
82      19 1, 20 3, 21 2, 22 2, 23 6, 24 5
83      25 5, 26 4, 27 5, 28 3, 29 7, 30 6
84      31 2, 32 3, 33 4, 34 5, 35 8, 36 9/;
85
86      VARIABLES
87      Z TOTAL COST OF ACQUIRED CELLS
88      X(I,K)
89      P(I,J,K)
90      N(I,J,K)
91
92      BINARY VARIABLES X,P,N;
93
94      EQUATIONS
95      OBJ      DEFINE OBJECTIVE FUNCTION
96      C1(K)    CONSTRAINT FOR NUMBER OF ACQUIRED CELLS

```

```

97    C2(I,J,K) CONSTRAINT FOR ALL BORDERS
98    C3(K)    CONSTRAINT FOR BORDER LENGTH
99    C4(I,J,K) CONSTRAINT FOR P&N MUTUAL EXCLUSION
100   C5(I,J,K) CONSTRAINT FOR PERIMETER BORDERS
101   C6(I)    CONSTRAINT FOR SUBREGIONS;
102
103   OBJ..
104       Z =E= SUM((I,K), C(I)*X(I,K));
105
106   C1(K)..
107       SUM(I,X(I,K)) =E= M(K);
108
109   C2(I,J,K) $ADJ(I,J)..
110       X(I,K) - X(J,K) - P(I,J,K) + N(I,J,K) =E= 0;
111
112   C3(K)..
113       SUM((I,J) $ADJ(I,J), P(I,J,K) + N(I,J,K)) =E= 2*L(K);
114
115   C4(I,J,K) $EXC(I,J)..
116       P(I,J,K) + N(I,J,K) =L= 1;
117
118   C5(I,J,K) $PER(I,J)..
119       X(I,K) - P(I,J,K) =E= 0;
120
121   C6(I)..
122       SUM(K, X(I,K)) =L= 1;
123
124   MODEL SUBREGION /ALL/;
125
126   OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 20000;
127
128   SOLVE SUBREGION USING MIP MINIMIZING Z;
129
130   OPTION MIP=ZOOM;
131
132   DISPLAY X.L,P.L,N.L;
133

```

### B.3 Partial Output File for Problem MSSA1

GAMS 2.20 VAX VMS

20-OCT-1991 19:22

GENERAL ALGEBRAIC MODELING SYSTEM  
COMPI LATION

```
1  SETS
2
3      I IMAGE ROWS WITH FRAME / 1 * 4/
4      J IMAGE COLS WITH FRAME / 1 * 4/;
5
6  TABLE C(I,J) COST MATRIX
7
8      1   2   3   4
9  1   6   5   3   4
10 2   1   3   2   2
11 3   5   4   1   3
12 4   2   2   3   1;
13
14 VARIABLES
15      Z TOTAL COST OF ALLOCATED CELLS
16      X(I,J) THE IMAGE CELL;
17
18 BINARY VARIABLE X;
19
20 EQUATIONS
21      COST OBJECTIVE FUNCTION
22      C1 CONSTRAINT FOR AREA
23      C2(I,J) CONSTRAINT FOR NEIGHBORS;
24
25 COST..
26      Z =E= SUM((I,J), C(I,J)*X(I,J));
27
28 C1..
29      SUM((I,J),X(I,J)) =E= 2;
30
31 C2(I,J)..
32      X(I-1,J) + X(I+1,J) + X(I,J-1) + X(I,J+1) - X(I,J) =G= 0;
33
34 MODEL SUBREGION /ALL/;
35
36 OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 50000;
37
38 OPTIONS LIMROW = 0, LIMCOL = 0;
39
40 SOLVE SUBREGION USING MIP MINIMIZING Z;
41
42 OPTION MIP=ZOOM;
43
44 DISPLAY X.L;
45
```

# MODEL STATISTICS

BLOCKS OF EQUATIONS	3	SINGLE EQUATIONS	18
BLOCKS OF VARIABLES	2	SINGLE VARIABLES	17
NON ZERO ELEMENTS	97	DISCRETE VARIABLES	16

GENERATION TIME = 0.310 SECONDS

EXECUTION TIME = 0.820 SECONDS

## S O L V E S U M M A R Y

MODEL	SUBREGION	OBJECTIVE	Z
TYPE	MIP	DIRECTION	MINIMIZE
SOLVER	ZOOM	FROM LINE	40

\*\*\*\* SOLVER STATUS 1 NORMAL COMPLETION

\*\*\*\* MODEL STATUS 1 OPTIMAL

\*\*\*\* OBJECTIVE VALUE 3.0000

RESOURCE USAGE, LIMIT 0.380 1000000.000

ITERATION COUNT, LIMIT 14 1000000

Z O O M / X M P --- Version 2.1 Oct 1988

Courtesy of Dr Roy E. Marsten,  
Department of Management Information Systems,  
University of Arizona,  
Tucson Arizona 85721, U.S.A.

No options file found - using defaults.

Work space needed (estimate) -- 102711 words.

Work space available -- 102711 words.

Iterations: Initial LP	14,	Time: 4.999971E-02
Heuristic	0,	0.000000E+00
Branch and bound	0,	0.000000E+00
Final LP	0,	0.000000E+00

\*\*\*\* REPORT SUMMARY :  
0 NONOPT  
0 INFEASIBLE  
0 UNBOUNDED

---- 44 VARIABLE X.L THE IMAGE CELL

3

2 1.000

3 1.000

# B:4 Input File for Problem MMSA2

GAMS 2.20 VAX VMS 21-OCT-1991 17:58  
 GENERAL ALGEBRAIC MODELING SYSTEM  
 COMPI LATION

```

1 SETS
2
3   I IMAGE ROWS WITH FRAME / 1*8/
4   J IMAGE COLS WITH FRAME / 1*8/
5   K SUBREGIONS / 1*2/;
6
7 TABLE C(I,J) COST MATRIX
8
9       1      2      3      4      5      6      7      8
10  1  100  100  100  100  100  100  100  100
11  2  100   6    5    3    4    3    4  100
12  3  100   2    5    1    8    5    3  100
13  4  100   7    7    3    4    9    2  100
14  5  100   1    3    2    2    6    5  100
15  6  100   5    4    5    3    7    6  100
16  7  100   2    3    4    5    8    9  100
17  8  100  100  100  100  100  100  100  100
18
19 PARAMETERS
20   M(K) CELLS
21   1 ^, 2 4/;
22
23 VARIABLES
24   Z TOTAL COST OF ALLOCATED CELLS
25   X(I,J,K) THE IMAGE CELL;
26
27 BINARY VARIABLE X;
28
29 EQUATIONS
30   COST OBJECTIVE FUNCTION
31   C1(K)      CONSTRAINT FOR SUBREGION AREA
32   C2(I,J,K)  CONSTRAINT FOR TWO FIRST ORDER NEIGHBORS
33   C3(I,J)    CONSTRAINT FOR ALLOCATION OF CELLS;
34
35 COST..
36   Z =E= SUM((I,J,K), C(I,J)*X(I,J,K));
37
38 C1(K)..
39   SUM((I,J),X(I,J,K)) =E= M(K);
40
41 C2(I,J,K)$ (C(I,J) NE 100)..
42   X(I-1,J,K) + X(I,J+1,K) + X(I+1,J,K) + X(I,J-1,K) - 2*X(I,J,K) =G=
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2
```



```

44 C3(I,J)..
45     SUM((K), X(I,J,K)) =L= 1;
46
47 MODEL SUBREGION /ALL/;
48
49 OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 50000;
50
51 OPTIONS LIMROW = 0, LIMCOL = 0;
52
53 SOLVE SUBREGION USING MIP MINIMIZING Z;
54
55 OPTION MIP=ZOOM;
56
57 DISPLAY X.L;
58

```

## B.5 Partial Output File for Problem MMSA4

GAMS 2.20 VAX VMS 20-OCT-1991 20:02  
GENERAL ALGEBRAIC MODELING SYSTEM  
COMPI LATION

```
1  SETS
2
3      I IMAGE ROWS WITH FRAME / 1*6/
4      J IMAGE COLS WITH FRAME / 1*6/
5      K SUBREGIONS / 1*2/;
6
7  TABLE C(I,J) COST MATRIX
8
9      1      2      3      4      5      6
10     1      6      5      3      4      3      4
11     2      2      5      1      8      5      3
12     3      7      7      3      4      9      2
13     4      1      3      2      2      6      5
14     5      5      4      5      3      7      6
15     6      2      3      4      5      8      9 ;
16
17  PARAMETERS
18      M(K) CELLS
19      / 1 9, 2 4/
20
21      W(K) WIDTH
22      / 1 3, 2 2/
23
24      H(K) HEIGHT
25      / 1 3, 2 2/;
26
27  VARIABLES
28      Z TOTAL COST OF ALLOCATED CELLS
29      X(I,J,K) THE IMAGE CELL
30      R(I,K) THE IMAGE ROW
31      L(J,K) THE IMAGE COLUMN;
32
33  BINARY VARIABLE X,R,L;
34
35  EQUATIONS
36      COST OBJECTIVE FUNCTION
37      C1(K)      CONSTRAINT FOR SUBREGION AREA
38      C2(I,J,K)  CONSTRAINT FOR NEIGHBORS
39      C3(I,J)     CONSTRAINT FOR ALLOCATION OF CELLS
40      C4(I,K)     CONSTRAINT TO LIMIT SUBREGION WIDTH
41      C5(K)       CONSTRAINT TO SET PROPER ROW (HEIGHT) VALUE
42      C6(J,K)     CONSTRAINT TO LIMIT SUBREGION HEIGHT
43      C7(K)       CONSTRAINT TO SET PROPER COLUMN (WIDTH) VALUE;
```

```

44
45 COST..
46     Z =E= SUM((I,J,K), C(I,J)*X(I,J,K));
47
48 C1(K)..
49     SUM((I,J),X(I,J,K)) =E= M(K);
50
51 C2(I,J,K)..
52     X(I-1,J,K) + X(I+1,J,K) + X(I,J-1,K) + X(I,J+1,K) -2*X(I,J,K) =G= 0;
53
54 C3(I,J)..
55     SUM((K), X(I,J,K)) =L= 1;
56
57 C4(I,K)..
58     SUM((J), X(I,J,K)) - W(K)*R(I,K) =E= 0;
59
60 C5(K)..
61     SUM((I), R(I,K)) - H(K) =E= 0;
62
63 C6(J,K)..
64     SUM((I), X(I,J,K)) - H(K)*L(J,K) =E= 0;
65
66 C7(K)..
67     SUM((J), L(J,K)) - W(K) =E= 0;
68
69 MODEL SUBREGION /ALL/;
70
71 OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 50000;
72
73 OPTIONS LIMROW = 0, LIMCOL = 0;
74
75 SOLVE SUBREGION USING MIP MINIMIZING Z;
76
77 OPTION MIP=ZOOM;
78
79 DISPLAY X.L;
80

```

#### MODEL STATISTICS

BLOCKS OF EQUATIONS	8	SINGLE EQUATIONS	139
BLOCKS OF VARIABLES	4	SINGLE VARIABLES	97
NON ZERO ELEMENTS	721	DISCRETE VARIABLES	96

GENERATION TIME = 1.030 SECONDS

EXECUTION TIME = 1.610 SECONDS

#### S O L V E S U M M A R Y

MODEL	SUBREGION	OBJECTIVE	Z
TYPE	MIP	DIRECTION	MINIMIZE
SOLVER	ZOOM	FROM LINE	75

\*\*\*\* SOLVER STATUS      1 NORMAL COMPLETION  
 \*\*\*\* MODEL STATUS      8 INTEGER SOLUTION  
 \*\*\*\* OBJECTIVE VALUE              43.0000

RESOURCE USAGE, LIMIT	137.380	1000000.000
ITERATION COUNT, LIMIT	9278	1000000

Z O O M / X M P      ---      Version 2.1 Oct 1988

Courtesy of Dr Roy E. Marsten,  
 Department of Management Information Systems,  
 University of Arizona,  
 Tucson      Arizona 85721, U.S.A.

No options file found - using defaults.

Work space needed (estimate)	--	121771 words.
Work space available	--	121771 words.

Note: The stopping tolerance is satisfied,  
 but the solution may not be optimal.

No solution better than      41.252638      can exist.  
 (absolute distance      1.7473621      , OPTCA      0.00000000E+00 )  
 (relative distance      4.2358E-02, OPTCR      0.1000      )

The branch and bound tree contained      203 nodes (max. 50000 nodes).

Iterations: Initial LP	184,	Time: 1.91000
Heuristic	49,	8.92999
Branch and bound	9044,	125.580
Final LP	1,	9.002686E-02

\*\*\*\* REPORT SUMMARY :      0      NONOPT  
                                  0      INFEASIBLE  
                                  0      UNBOUNDED

-----      79 VARIABLE      X.L      THE IMAGE CELL

	1	2
1.2		1.000
1.3		1.000
2.2		1.000
2.3		1.000

4.1	1.000
4.2	1.000
4.3	1.000
5.1	1.000
5.2	1.000
5.3	1.000
6.1	1.000
6.2	1.000
6.3	1.000

## B.6 Partial Output File for Problem MSSA3

GAMS 2.20 VAX VMS

20-OCT-1991 18:04

GENERAL ALGEBRAIC MODELING SYSTEM  
COMPI LATION

```
1  SETS
2      I IMAGE ROWS WITH FRAME / 1*6/
3      J IMAGE COLS WITH FRAME / 1*6/
4      K SUBREGIONS / 1*1/;
5
6  TABLE C(I,J) COST MATRIX
7      1   2   3   4   5   6
8  1   1   1   7   7   1   1
9  2   1   1   2   2   1   1
10 3   7   2   3   3   2   7
11 4   7   2   3   3   2   7
12 5   1   1   2   2   1   1
13 6   1   1   7   7   1   1;
14
15 PARAMETERS
16     M(K) CELLS
17 / 1 16/
18
19     W(K) WIDTH
20 / 1 4/
21
22     H(K) HEIGHT
23 /1 4/;
24
25 VARIABLES
26     Z TOTAL COST OF ALLOCATED CELLS
27     X(I,J,K) THE IMAGE CELL
28     R(I,K) THE IMAGE ROW
29     L(J,K) THE IMAGE COLUMN;
30
31 BINARY VARIABLE X,R,L;
32
33 EQUATIONS
34     COST          OBJECTIVE FUNCTION
35     C1(K)         SUBREGION CONSTRAINT FOR AREA
36     C2(I,J)       CONSTRAINT FOR ALLOCATION OF CELLS
37     C3(I,K)       CONSTRAINT TO LIMIT SUBREGION WIDTH
38     C4(K)         CONSTRAINT TO SET PROPER ROW (HEIGHT) VALUE
39     C5(J,K)       CONSTRAINT TO LIMIT SUBREGION HEIGHT
40     C6(K)         CONSTRAINT TO SET PROPER COLUMN (WIDTH) VALUE
41     C7(I,K)       ROW ADJACENCY CONTIGUITY CONSTRAINT
42     C8(J,K)       COLUMN ADJACENCY CONTIGUITY CONSTRAINT;
43
44 COST..
```

```

45      Z =E= SUM((I,J,K), C(I,J)*X(I,J,K));
46
47      C1(K)..
48          SUM((I,J), X(I,J,K)) =E= M(K);
49
50      C2(I,J)..
51          SUM((K), X(I,J,K)) =L= 1;
52
53      C3(I,K)..
54          SUM((J), X(I,J,K)) - W(K)*R(I,K) =E= 0;
55
56      C4(K)..
57          SUM((I), R(I,K)) - H(K) =E= 0;
58
59      C5(J,K)..
60          SUM((I), X(I,J,K)) - H(K)*L(J,K) =E= 0;
61
62      C6(K)..
63          SUM((J), L(J,K)) - W(K) =E= 0;
64
65      C7(I,K)..
66          R(I-1,K) + R(I-2,K) + R(I-3,K) + R(I+1,K) + R(I+2,K) + R(I+3,K)
67          - 3*R(I,K) =G= 0;
68
69      C8(J,K)..
70          L(J-1,K) + L(J-2,K) + L(J-3,K) + L(J+1,K) + L(J+2,K) + L(J+3,K)
71          = 3*L(J,K) =G= 0;
72
73      MODEL SUBREGION /ALL/;
74
75      OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 50000;
76
77      OPTIONS LIMROW = 0, LIMCOL = 0;
78
79      SOLVE SUBREGION USING MIP MINIMIZING Z;
80
81      OPTION MIP=ZOOM;
82
83      DISPLAY X.L;

```

#### MODEL STATISTICS

BLOCKS OF EQUATIONS	9	SINGLE EQUATIONS	64
BLOCKS OF VARIABLES	4	SINGLE VARIABLES	49
NON ZERO ELEMENTS	265	DISCRETE VARIABLES	48
GENERATION TIME	=	0.490 SECONDS	
EXECUTION TIME	=	0.940 SECONDS	

# S O L V E      S U M M A R Y

MODEL	SUBREGION	OBJECTIVE	Z
TYPE	MIP	DIRECTION	MINIMIZE
SOLVER	ZOOM	FROM LINE	79

\*\*\*\* SOLVER STATUS      1 NORMAL COMPLETION  
 \*\*\*\* MODEL STATUS      8 INTEGER SOLUTION  
 \*\*\*\* OBJECTIVE VALUE              32.0000

RESOURCE USAGE, LIMIT	160.330	1000000.000
ITERATION COUNT, LIMIT	24709	1000000

Z O O M / X M P      ---      Version 2.1 Oct 1988

Courtesy of Dr Roy E. Marsten,  
 Department of Management Information Systems,  
 University of Arizona,  
 Tucson      Arizona 85721, U.S.A.

No options file found - using defaults.

Work space needed (estimate)	--	108539 words.
Work space available	--	108539 words.

Note: The stopping tolerance is satisfied,  
 but the solution may not be optimal.

No solution better than      29.200000      can exist.  
 (absolute distance      2.8000000      , OPTCA      0.00000000E+00 )  
 (relative distance      9.5890E-02, OPTCR      0.1000      )

The branch and bound tree contained      539 nodes (max. 50000 nodes).

Iterations: Initial LP	69,	Time: 0.429993
Heuristic	37,	1.51001
Branch and bound	24602,	157.860
Final LP	1,	7.000732E-02

\*\*\*\* REPORT SUMMARY :      0      NONOPT  
                                  0      INFEASIBLE  
                                  0      UNBOUNDED

-----      83 VARIABLE      X.L              THE IMAGE CELL

1

2.2	1.000
2.3	1.000



2.4	1.000
2.5	1.000
3.2	1.000
3.3	1.000
3.4	1.000
3.5	1.000
4.2	1.000
4.3	1.000
4.4	1.000
4.5	1.000
5.2	1.000
5.3	1.000
5.4	1.000
5.5	1.000

## Appendix C. *Programs and Files for The Network Formulations of the Subregion Allocation Problems*

### C.1 *The "SUBNETS" Pascal Program*

```
program subnets(input, output, rcdata, optdata, image, working, nsa);

{ Program to read in the data file,
  query the user for information,
  and create the working file }

type bigness = array[1..5] of integer;

var
  working: file of integer;
  rcdata, optdata, image, nsa: text;
  rows, columns, size, gmin, gmax, option, subs, choice: integer;
  area: bigness;

{***** PROCEDURE INTRO *****)

procedure intro;

begin {intro}
  writeln ('*****');
  writeln ('      THE USE OF NETWORKS FOR SUBREGION ALLOCATION:');
  writeln ('      BASED ON CONCEPTS DEVELOPED BY BENABDALLAH AND WRIGHT');
  writeln ('      AND IMPLEMENTATION PROCEDURES DEVELOPED BY BURKE. ');
  writeln ('      CREATED BY THOMAS G. REED');
  writeln ('*****');
  writeln;
  writeln ('This program will use data contained in the text file');
  writeln ('"IMAGE.DAT". The SAS portion of the program will create');
  writeln ('for later execution the SAS file "NSA" (which stands for');
  writeln ('"Network Subregion Allocation"). ');
  writeln;
  writeln ('PURPOSE');
  writeln ('The purpose of this program is twofold:');
  writeln;
  writeln (' (1) determine which pixels of an image make up the');
  writeln ('      subregion(s) of interest as specified by the spectral');
  writeln ('      and spatial requirements of the program user;');
  writeln;
  writeln ('      OR');
  writeln;
  writeln (' (2) determine the minimum cost single subregion of a data matrix');
```

```

writeln ('      representing cost, population, grey values, distance, etc.;');
writeln;
writeln ('INPUT');
writeln ('The data matrix MUST be contained in a TEXT file called IMAGE.DAT.');
```

writeln ('The user will be queried for additional information as needed.');

```

writeln;
writeln ('OUTPUT');
writeln ('The output of this program depends on the options chosen by the');
writeln ('user. Working files used for data manipulation and either a GAMS');
writeln ('or SAS input file are common outputs for all options.');
```

writeln;

```

end; {intro}

{***** PROCEDURE QUERY_USER *****/}

procedure query_user(var rows, columns, size, gmin, gmax, option, choice: integer;
                    var area: bigness);

var
    i, j: integer;
    cell: array [1..100, 1..100] of integer;    {temp storage of data values}

begin {query_user}
    option := 0;
    choice := 0;
    size := 0;
    rows := 0;
    columns := 0;
    writeln ('*****');
    while ( (rows<1) or (rows>20) or (columns<1) or (columns>20) ) do
    begin
        writeln ('NOTE: the image cannot exceed 20 cells by 20 cells, thank you.');
```

writeln;

```

        writeln ('Enter the number of ROWS of the image/data');
        readln (rows);
        writeln ('Enter the number of COLUMNS of the image/data');
        readln (columns)
    end; {while rows & columns}
    writeln;
    writeln (rcdata, rows);
    writeln (rcdata, columns);
    size := rows*columns;
    writeln;
    writeln ('*****');
    writeln ('ORIGINAL DATA MATRIX:');           {show user what data looks like}
    writeln;
    for i := 1 to rows do
    begin
        for j := 1 to columns do
        begin
            read (image, cell[i,j]);    {read in data}
```

```

        write (cell[i,j]:4)           {print to screen in 2-D format}
    end;
    writeln;
end;
writeln;
writeln ('*****');
writeln;

writeln ('# of rows is:', rows:3, ' ; columns is:', columns:3, ' ; size is:', size:4);
writeln;
writeln;
while (option < 1) or (option > 2) do      {query user for program option}
begin
    writeln ('      *****');
    writeln ('Enter one of the following program options:');
    writeln;
    writeln ('1:  IMAGE SUBREGION DETERMINATION');
    writeln ('      (based on spatial and spectral requirements)');
    writeln;
    writeln ('2:  MIN-COST SUBREGION DETERMINATION');
    writeln;
    readln (option)
end; {while option}
writeln (optdata, option);                {store user's option selection}

case option of
    1 : begin                                {query user for GMIN and GMAX}
        gmin := 1;
        gmax := 0;
        while ( (gmin>gmax) or (gmin<0) or (gmin>255) or (gmax<0) or (gmax>255) )do
        begin
            writeln ('      *****');
            writeln;
            writeln ('NOTE: The subregion minimum grey value you are about to enter');
            writeln ('must not exceed the subregion maximum grey value. ');
            writeln ('Also, both must be between 0 and 255, inclusive. ');
            writeln;
            writeln ('Enter the MINIMUM grey value of the subregion of interest');
            readln (gmin);
            writeln ('Enter the MAXIMUM grey value of the subregion of interest');
            readln (gmax)
        end {while}
    end; {option 1}
    2 : begin                                {query for subregions and areas of each}
        subs := 0;
        for i := 1 to 5 do area[i] := 0;
        while ( (subs<1) or (subs>5) ) do
        begin
            writeln ('      *****');
            writeln;
            writeln ('NOTE: The maximum number of subregions that can be');

```

```

        writeln ('determined is 5. ');
        writeln;
        writeln ('Enter the number of subregions you wish to determine');
        readln (subs)
    end; {while}

    for i := 1 to subs do
    begin
        while ( (area[i]<1) or (area[i]>size) or (area[i]>15) ) do
        begin
            writeln ('      *****');
            writeln;
            writeln ('NOTE: The desired subregion size you are about to enter must');
            writeln ('not exceed 15. ');
            writeln;
            writeln ('Enter the desired subregion size for subregion #', i:2);
            readln (area[i])
        end {while area ...}
    end; {for i}
    writeln;
    choice:= 1;                {compactness is an option in the next program}
    writeln (optdata, choice);    {store choice selection}
    writeln (optdata, subs);      {store the number of subregions}
    for i := 1 to subs do writeln (optdata, area[i]) {store subregion areas}
    end {option 2}
end {case}
end; {query_user}

{***** PROCEDURE OPT1_WORK_FILE *****)}

procedure opt1_work_file(var rows, columns, gmin, gmax, choice: integer);

var
    i, j, area: integer;
    pixel: array [1..100, 1..100] of integer;

begin {opt1_work_file}

    choice := 0;                {set some initial values for looping later}
    area := 0;                  {initial subregion(s) area}

    for j := 1 to columns+2 do    { Create image frame }
    begin
        pixel[1,j] := 88;
        pixel[rows+2,j] := 88
    end;
    for i := 2 to rows+1 do
    begin
        pixel[i,1] := 88;
        pixel[i,columns+2] := 88
    end;

```

```

for i := 2 to rows+1 do
    { Establish new image pixel values }
begin
    for j := 2 to columns+1 do
    begin
        read (image,pixel[i,j]);
        if (pixel[i,j] >= gmin) and (pixel[i,j] <= gmax)
        then
            begin
                pixel[i,j] := 1;
                area := area+1
            end {if then}
        else pixel[i,j] := 88
        end
    end;
end;

writeln;
writeln ('*****');
writeln ('WORKING IMAGE WITH "FRAME":');
writeln;
for i := 1 to rows+2 do
begin
    for j := 1 to columns+2 do
    begin
        write (working,pixel[i,j]);
        { Create working file }
        write (pixel[i,j]:3)
        {show user working file}
    end;
    writeln;
end;
writeln;
writeln ('NOTE: Pixels denoted with a 1 represent subregion candidate');
writeln ('pixels based on your spectral requirements. ');
writeln;
writeln ('The candidate pixels comprise a total area of: ', area:5);
writeln ('*****');

writeln;
while (choice < 1) or (choice > 5) do {query user for comp/cont option}
begin
    writeln ('Enter one of the following compactness/contiguity options:');
    writeln ('
        *****');
    writeln ('1:  None: no compactness/contiguity requirements');
    writeln ('      (show all pixels within the grey range of interest)');
    writeln ('2:  Marginal compactness/contiguity requirements');
    writeln ('      (delete single pixel renegades)');
    writeln ('      (pixels must have at least one 1st- or 2nd-order neighbor)');
    writeln ('3:  Low compactness/contiguity requirements');
    writeln ('      (pixels must have at least one 1st-order neighbor)');
    writeln ('4:  Medium compactness/contiguity requirements');
    writeln ('      (pixels must have at least two total neighbors)');
    writeln ('5:  High compactness/contiguity requirements');

```

```

        writeln ('      (pixels must have at least two 1st-order neighbors)');
        writeln;
        readln (choice)
    end; {while}
    write (optdata, choice)      {store user's choice selection}
end; {opt1_work_file}

{***** PROCEDURE OPT2_WORK_FILE *****)}

procedure opt2_work_file(var rows, columns: integer);

var
    i, j: integer;
    cell: array [1..100, 1..100] of integer;

begin {opt2_working_file}

    for j := 1 to columns+2 do                { Create matrix frame }
    begin
        cell[1,j] := 888;
        cell[rows+2,j] := 888
    end;
    for i := 2 to rows+1 do
    begin
        cell[i,1] := 888;
        cell[i,columns+2] := 888
    end;

    for i := 2 to rows+1 do                    { Establish work file data }
    begin
        for j := 2 to columns+1 do
        begin
            read (image, cell[i,j])
        end
    end;

    writeln;
    writeln ('*****');
    writeln ('WORKING DATA MATRIX WITH "FRAME":');
    writeln;
    for i := 1 to rows+2 do
    begin
        for j := 1 to columns+2 do
        begin
            write (working, cell[i,j]);          { Create working file }
            write (cell[i,j]:4)                  {show user working file}
        end;
        writeln;
    end;
    writeln;
    writeln ('*****');

```

```

end; {opt2_work_file}

{***** PROCEDURE GAMS_FILE *****}

procedure gams_file(var rows, columns, choice: integer);

var
  i,j: integer;
  pixel: array [1..100, 1..100] of integer;

begin {gams_file}                                {create the GAMS file NSA.DAT}

  writeln (nsa, 'SETS');
  writeln (nsa);
  writeln (nsa, '      I image rows with frame / 1 *', (rows+2):3, '/');
  writeln (nsa, '      J image cols with frame / 1 *', (columns+2):3, '/');
  writeln (nsa);
  writeln (nsa, 'TABLE C(I,J) cost matrix');
  write (nsa, '      1');                                {label for column 1}
  for j := 2 to columns+2 do                            {labels for columns 2 and up}
  begin
    write (nsa, j:3)
  end;
  writeln (nsa);
  for i := 1 to rows+2 do                                {read in working file}
  begin
    for j := 1 to columns+2 do
    begin
      read (working, pixel[i,j])
    end
  end;
  for i := 1 to rows + 1 do                                {write image matrix into NSA.DAT file}
  begin
    write (nsa, i:2);                                {labels for row numbers}
    for j := 1 to columns+2 do
    begin
      write (nsa, pixel[i,j]:3)                        {grey scale data from WORKING file}
    end;
    writeln (nsa);
  end;
  i := rows+2;                                            {last row written last so that ... }
  write (nsa, i:2);
  for j := 1 to columns+2 do
  begin
    write (nsa, pixel[i,j]:3)
  end;
  writeln (nsa, ',');                                {semicolon is added}
  writeln (nsa);
  writeln (nsa, 'VARIABLES');
  writeln (nsa, '      Z total area of allocated cells');

```



```

writeln (nsa, '    X(I,J) the image cell;');
writeln (nsa);
writeln (nsa, 'BINARY VARIABLE X;');
writeln (nsa);
writeln (nsa, 'EQUATIONS');
writeln (nsa, '    SIZE  OBJECTIVE FUNCTION');
write (nsa, '    C1(I,J)    CONSTRAINT FOR NONSUBREGION ALLOCATION');
case choice of
  1 : writeln (nsa, ','); {choice 1 does not require a contiguity constraint}
  2,3,4,5 : begin
        writeln (nsa);
        writeln (nsa, '    C2(I,J)    CONSTRAINT FOR CONTIGUITY;')
      end {option 2,3}
end; {case}
writeln (nsa);
writeln (nsa, 'SIZE..');
writeln (nsa, '    Z =E= SUM((I,J), X(I,J));');
writeln (nsa);
writeln (nsa, 'C1(I,J)$(C(I,J) NE 1)..');
writeln (nsa, '    X(I,J) =E= 0;');
case choice of
  2 : begin
        writeln (nsa);
        writeln (nsa, 'C2(I,J)$(C(I,J) EQ 1)..');
        writeln (nsa, '    X(I-1,J-1) + X(I-1,J) + X(I-1,J+1) + X(I,J+1)
          + X(I+1,J+1) + X(I+1,J) + X(I+1,J-1) + X(I,J-1) - X(I,J) =G= 0;')
      end; {option 2}
  3 : begin
        writeln (nsa);
        writeln (nsa, 'C2(I,J)$(C(I,J) EQ 1)..');
        writeln (nsa, '    X(I-1,J) + X(I,J+1) + X(I+1,J) + X(I,J-1) - X(I,J) =G= 0;')
      end; {option 3}
  4 : begin
        writeln (nsa);
        writeln (nsa, 'C2(I,J)$(C(I,J) EQ 1)..');
        writeln (nsa, '    X(I-1,J-1) + X(I-1,J) + X(I-1,J+1)
          + X(I,J+1) + X(I+1,J+1) + X(I+1,J) + X(I+1,J-1) + X(I,J-1) - 2*X(I,J) =G= 0;')
      end; {option 4}
  5 : begin
        writeln (nsa);
        writeln (nsa, 'C2(I,J)$(C(I,J) EQ 1)..');
        writeln (nsa, '    X(I-1,J) + X(I+1,J) + X(I,J-1) + X(I,J+1) - 2*X(I,J) =G= 0;')
      end {option 3}
end; {case}
writeln (nsa);
writeln (nsa, 'MODEL SUBREGION /ALL/;');
writeln (nsa);
writeln (nsa, 'OPTIONS ITERLIM = 1000000, RESLIM = 1000000, WORK = 50000;');
writeln (nsa);
writeln (nsa, 'OPTIONS LIMROW = 0, LINCOL = 0;');
writeln (nsa);

```

```

        writeln (nsa, 'SOLVE SUBREGION USING MIP MAXIMIZING Z;');
        writeln (nsa);
        writeln (nsa, 'OPTION MIP=ZOOM;');
        writeln (nsa);
        writeln (nsa, 'DISPLAY X.L;');
        writeln (nsa);
        writeln;
        writeln ('***** FINDING OPTIMUM SOLUTION . . . PLEASE WAIT *****');
        writeln;
    end; {gams_file}

{***** MAIN PROGRAM : SUBREGION *****}

begin {subnets}
    intro;
    reset (image);
    rewrite (rcdata);
    rewrite (optdata);
    query_user (rows, columns, size, gmin, gmax, option, choice, area); {QUERY_USER proc}
    reset (image);
    rewrite (working);
    case option of
        1 : begin
            opt1_work_file (rows, columns, gmin, gmax, choice);
            end; {option 1}
        2 : begin
            opt2_work_file (rows, columns);
            end; {option 2}
    end; {case}
    reset (working);
    rewrite (nsa);
    case option of
        1 : gams_file (rows, columns, choice);
        2 : begin
            {run GAMS_FILE proc}
            {instruct user to run SAS_FILE prog}
            writeln;
            writeln;
            writeln ('Type "run [sas filename]" (without quotes) at the prompt;
                    press [RETURN]');
            writeln;
            end {option 2}
    end {case}
end. {subnets}

```

## C.2 The "AREAWALL" Pascal Program

```
program areawall(input, output, rcdata, optdata, arcname, celldata, working, nsa);

{ The program SUBNETS must be run first in order to create the working data file
  and query the user for necessary information used in this program. }

{ This program creates the SAS file used for the AREAWALL network-with-side-
  constraints implementation of single and multiple subregion allocation problems. }

{ the row/column format for representing the working data matrix is
  used for the calculations; a return to the sequentially numbered matrix cells
  is used for node and arc names since they cannot exceed 8 characters in length }

{ the _space variables are used to determine the width needed to write the data }

{ The following is an explanation of the files and what they are used for:
  rcdata: holds the values of the "rows" and "columns" variables of the matrix;
  optdata: holds the values of the user's options, choices, subregion(s), area(s);
  arcname: holds the list of arcnames;
  celldata: holds the number of the cells used in each node and arc;
  working: holds the working file cost matrix;
  nsa: the sas input file which this program ultimately creates.
  Note: rcdata, optdata, and working were created by the subnets program.
  Note: arcname and celldata are utilized in the writing of side constraints. }

type string = packed array [1..50, 1..8] of char;
   bigness = array[1..5] of integer;

var
  working: file of integer;
  rcdata, optdata, arcname, celldata, nsa: text;
  bbarc: string;           {allows several arcs to have flows set}
  bbcopy: string;          {used for determining position of arc names}
  wall: array [1..26] of char; {a..z -- allows 26-cell subregion sizes}
  i, j, k, lineup, cellnum, cellspace, rmicell, rmispace, rpicell, rpispace,
  cmicell, cmispace, cpicell, cpispace, cellcopy, rows, columns,
  size, gmin, gmax, option, choice, subs, maxarea,
  selection, totarcs, setarcs, nomatch: integer;
  area: bigness;            {subregion sizes}
  counter: array [1..50] of integer; {for finding arc names in the arcname file}
  cell: array [1..100, 1..100] of integer;
  arcch: char;
  sidecon, bbset, match: boolean; { for side constraints }
                                   {bbset indicates whether the user wants to set flow in an arc}

begin {areawall}
  reset (rcdata);
  reset (optdata);
  reset (working);

  { the following section establishes the possible wall values in the wall array }
```

```
{ Note: the number of walls is determined by the size of the largest subregion. }
```

```
wall[1] := 'a';
for i := 2 to 26 do wall[i] := chr(ord(wall[i-1])+1);
```

```
maxarea := 0;
read (optdata, option);
read (optdata, choice);
read (optdata, subs);
for i := 1 to subs do
begin
  read (optdata, area[i]);
  if area[i]>maxarea then maxarea:= area[i] {get subregion area info}
  {store largest subregion size}
end; {for i}
sidecon := true;
bbset := false;
read (rcdata, rows);
read (rcdata, columns);
size := (rows)*(columns);
```

```
{ total network arcs = the total number of cell neighbors (not including the frame) }
{ for each maxarea interwall space (2 for each corner, 3 for each border, 4 for each
remaining cell, and 2*size for source and sink arcs), plus the source and
destination arcs (size + subs*size) }
```

```
totarcs := (((4*size)-(2*rows)-(2*columns))*(maxarea-1)) + ((subs+1)*size);
```

```
rewrite (nsa);
rewrite (arcname);
rewrite (celldata); {store cell numbers in arcname order for compact. constraints}
```

```
for i := 1 to rows+2 do {read in working file}
begin
  for j := 1 to columns+2 do
  begin
    read (working, cell[i,j])
  end
end;
```

```
{ create SAS file NSA.DAT composed of node data, arc data, side constraint data }
{ Flow from the source equals the number of subregions; there is one
destination (d1, d2, ... d(subs)) for each subregion. }
{ Write all arc names to the arcname file as they are created. }
{ The current cell number will be stored in the file celldata }
{ the dummy 8888 number is used since we only want arcs OUT OF nodes }
{ for some side constraints; i.e., source to wall "a" arcs are not OUT OF nodes }
```

```
{ periods represent coefficients of 0 }
```

```
writeln (nsa, 'data noded;');
writeln (nsa, ' input _node_$ _sd;');
writeln (nsa, ' cards;');
writeln (nsa, 'ss', subs:3);
```

```

for i:= 1 to subs do writeln (nsa, 'd',i:1, ' -1');
writeln (nsa, ',');
writeln (nsa, 'data arcd;');
writeln (nsa, 'input _from_$ _to_$ _cost_ _capac_ _name_$;');
writeln (nsa, 'cards;');

{ write the arc data from the source to the first wall }
{ source will not flow into frame cells since they will not be in the network }

cellspace := 1; {cell print space}
for i := 2 to rows+1 do
begin
  for j := 2 to columns+1 do
  begin
    cellnum := (((i-1)*(columns+2))+j);
    if cellnum > 9 then cellspace := 2;
    if cellnum > 99 then cellspace := 3;
    write (nsa, 'ss ', 'a',cellnum:cellspace);
    writeln (nsa, cell[i,j]:4, ' 1 ssa',cellnum:cellspace);
    writeln (arcname, 'ssa',cellnum:cellspace);
    writeln (celldata, '8888') {dummy number for maintaining position}
  end {for j}
end; {for i}

{ write the arc data from the first wall to the last wall (based on value of maxarea) }
{ (from each cell in a wall to its matrix neighbors in the next) }
{ NOTE: arcs to the frame cells (nodes) are not needed; frame nodes }
{ will not exist in the final network. They simply make the determination }
{ of neighbors a simpler task. }

for k := 1 to maxarea-1 do
begin
  cellspace := 1; {reset cell print space}
  rmispace := 1; {cell minus 1 row print space--for 'next' wall}
  rpispace := 1; {cell plus 1 row print space--for 'next' wall}
  cmispace := 1; {cell minus 1 column print space--for 'next' wall}
  cpispace := 1; {cell plus 1 column print space--for 'next' wall}
  for i := 2 to rows+1 do
  begin
    for j := 2 to columns+1 do
    begin
      cellnum := (((i-1)*(columns+2))+j); {determine the cell being used}
      rmicell := (((i-2)*(columns+2))+j); {determine the neighbors of the cell}
      rpicell := ((i*(columns+2))+j); {row-1, row+1, col-1, col+1}
      cmicell := (((i-1)*(columns+2))+j-1);
      cpicell := (((i-1)*(columns+2))+j+1);

      if cellnum > 9 then cellspace := 2, {determine the space needed to}
      if cellnum > 99 then cellspace := 3; {print the cell number}
      if rmicell > 9 then rmispace := 2;
      if rpicell > 99 then rmispace := 3;

```

```

if rp1cell > 9 then rp1space := 2;
if rp1cell > 99 then rp1space := 3;
if cm1cell > 9 then cm1space := 2;
if cm1cell > 99 then cm1space := 3;
if cp1cell > 9 then cp1space := 2;
if cp1cell > 99 then cp1space := 3;

{write to file NSA the arc data from the cell to its neighbors for each maxarea-1 wall}
{write all arc names to the arcname file}
{ do not include nodes or arcs for frame cells (cells with cost of 888) }

    if (cell[i-1,j]<>888) then
    begin
        write (nsa, wall[k],cellnum:cellspace,' ',wall[k+1],rm1cell:rm1space);
        write (nsa,' ',cell[i-1,j]:4,' 1 ');
        writeln (nsa, wall[k],cellnum:cellspace,wall[k+1],rm1cell:rm1space);
        writeln (arcname, wall[k],cellnum:cellspace,wall[k+1],rm1cell:rm1space);
        writeln (celldata, cellnum)
    end; {if cell[i-1,j]}

    if (cell[i,j+1]<>888) then
    begin
        write (nsa, wall[k],cellnum:cellspace,' ',wall[k+1],cp1cell:cp1space);
        write (nsa,' ',cell[i,j+1]:4,' 1 ');
        writeln (nsa, wall[k],cellnum:cellspace,wall[k+1],cp1cell:cp1space);
        writeln (arcname, wall[k],cellnum:cellspace,wall[k+1],cp1cell:cp1space);
        writeln (celldata, cellnum)
    end; {if cell[i,j+1]}

    if (cell[i+1,j]<>888) then
    begin
        write (nsa, wall[k],cellnum:cellspace,' ',wall[k+1],rp1cell:rp1space);
        write (nsa,' ',cell[i+1,j]:4,' 1 ');
        writeln (nsa, wall[k],cellnum:cellspace,wall[k+1],rp1cell:rp1space);
        writeln (arcname, wall[k],cellnum:cellspace,wall[k+1],rp1cell:rp1space);
        writeln (celldata, cellnum)
    end; {if cell[i+1,j]}

    if (cell[i,j-1]<>888) then
    begin
        write (nsa, wall[k],cellnum:cellspace,' ',wall[k+1],cm1cell:cm1space);
        write (nsa,' ',cell[i,j-1]:4,' 1 ');
        writeln (nsa, wall[k],cellnum:cellspace,wall[k+1],cm1cell:cm1space);
        writeln (arcname, wall[k],cellnum:cellspace,wall[k+1],cm1cell:cm1space);
        writeln (celldata, cellnum)
    end {if cell[i,j-1]}
end {for j}
end {for i}
end; {for k}

{ write the arc data from the appropriate walls to their respective destinations }

```

```
{ again, frame cells are not included since they will not be in the network }
```

```
for k := 1 to subs do
begin
cellspace := 1;                                {reset cell print space}
for i := 2 to rows+1 do
begin
for j := 2 to columns+1 do
begin
cellnum := ((i-1)*(columns+2))+j);
if cellnum > 9 then cellspace := 2;
if cellnum > 99 then cellspace := 3;
write (nsa, wall[area[k]],cellnum:cellspace,' d',k:1,' . 1 ');
writeln (nsa, wall[area[k]],cellnum:cellspace,'d',k:1);
writeln (arcname, wall[area[k]],cellnum:cellspace,'d',k:1);
writeln (celldata, cellnum)
end {for j}
end {for i}
end; {for k}
writeln (nsa, ',');
```

```
{ SIDE CONSTRAINTS SECTION }
```

```
choice:= 0;                                     {reset to allow for user changes}
writeln;
writeln ('*****');
while ( (choice<1) or (choice>2) ) do
begin
writeln ('Enter one of the following compactness/contiguity options:');
writeln (' *****');
writeln ('1:  None: no compactness/contiguity requirements');
writeln ('      ( MUST be chosen if any subregion area is 3 or smaller');
writeln ('      or equal to 5 )');
writeln ('      (all cells will, however, have at least one 1st-order');
writeln ('      neighbor)');
writeln ('2:  High compactness/contiguity requirements');
writeln ('      (cells must have at least two 1st-order neighbors)');
writeln;
readln (choice)
end; {while choice}
```

```
selection := 0;
writeln;
writeln ('*****');
while ( (selection<1) or (selection>2) ) do
begin
writeln ('Select the option below which describes your situation:');
writeln;
writeln ('1: You are attempting to solve the problem for the first time');
writeln ('      (or, if not, you have no arcs on which you wish to set the flow).');
writeln;
```

```

        writeln ('2: You need to set the flow on one or more of the arcs to 1. ');
        writeln ('      (i.e., you have already solved the problem but have split ');
        writeln ('      flows to contend with via a branch-and-bound method) ');
        writeln;
        readln (selection);
    end; {while selection}
    if selection = 2 then
    begin
        setarcs := 0;
        bbset := true;
        writeln;
        writeln ('How many arcs need to have their flow set to 1? ');
        while ( (setarcs < 1) or (setarcs > totarcs) ) do
        begin
            writeln ('(the number must be greater than 0) ');
            writeln (' and probably should not exceed ', (maxarea+1):4, ' ');
            writeln;
            readln (setarcs)
        end {while setarcs}
    end; {if selection}

    if ( (maxarea < 2) and (not bbset) ) then sidecon := false; {no side constraints needed}

    if sidecon then
    begin
        { list arc variables since the side constraints are in terms of the arcs}

        writeln (nsa, 'data cond; ');
        writeln (nsa, 'input ');

        reset (arcname);

        j := 1;
        for i := 1 to totarcs do
        begin
            bbcopy[1] := '          ';
            readln (arcname, bbcopy[1]);
            write (nsa, bbcopy[1], ' ');
            j := j+1;
            if j > 10
            then begin
                j := 1;
                writeln (nsa)
            end {if j, then}
        end; {for i}

        writeln (nsa, '_type_$ _rhs_ ');
        writeln (nsa, 'cards ');

        { write the limit-cells-to-1-subregion side constraints }
    end;

```



{ 'lineup' keeps the lines from exceeding 40 characters in length }

```

for i := 2 to rows+1 do
begin
  for j := 2 to columns+1 do
  begin
    reset (celldata);
    lineup := 1;
    cellnum := (((i-1)*(columns+2))+j);    {determine the current cell}

    for k := 1 to totarcs do {search thru entire celldata file}
    begin
      readln (celldata, cellcopy); {get stored cellnum from file celldata}
      if cellcopy = cellnum        {if we are at the current cell number ...}
      then write (nsa, '1 ')      {write 1}
      else write (nsa, '. ');    {if not, write "."; not flow out of the cell}
      lineup := lineup+1;
      if lineup>40 then
      begin
        writeln (nsa);
        lineup := 1
      end {if lineup}
    end; {for k}

    writeln (nsa, 'LE 1');
    writeln (nsa)
  end {for j}
end; {for i}

```

{ write the compactness constraints, if needed }

{ NOTE: since the compactness constraints are in terms of the flow to/from nodes, and are not based on the subregion(s), then if any subregion area is 3 or 5, NO compactness constraints can be written. The user's compactness choice will be changed if need be. }

```

for i:= 1 to subs do if ( (area[i]=3) or (area[i]=5) ) then choice := 1;

```

```

{ -2 is the coefficient for all arcs out of the current cell }
{ and a 1 is the coefficient for all arcs out of its neighbors; }
{ all other arcs will have coefficients of 0 }
{ the file celldata is searched for cell and neighbor matches }
{ to know where to properly place the coefficients }

```

```

if (choice=2) then
begin
  for i := 2 to rows+1 do
  begin
    for j := 2 to columns+1 do
    begin
      reset (celldata);

```

```

lineup := 1;
cellnum := (((i-1)*(columns+2))+j);      {determine the current cell}
rmicell := (((i-2)*(columns+2))+j);      {determine the neighbors ...}
rpcell := ((i*(columns+2))+j);
cmicell := (((i-1)*(columns+2))+j-1));
cpicell := (((i-1)*(columns+2))+j+1));

for k := 1 to totarcs do {search thru entire celldata file}
begin
  readln (celldata, cellcopy); {get stored cellnum from file celldata}
  if cellcopy = cellnum        {if we are at the current cell number ...}
  then begin
    write (nsa, '-2 ');
    lineup := lineup+2
  end {if cellcopy, then}
  else begin {else, if we are at a neighbor cell ... }
    if ( (cellcopy=rmicell) or (cellcopy=rpcell) or (cellcopy=cmicell)
      or (cellcopy=cpicell) )
    then write (nsa, '1 ')
    else write (nsa, '. '); {not the cell nor a neighbor of the cell}
    lineup := lineup+1
  end; {if cellcopy, else}
  if lineup>40 then
  begin
    writeln (nsa);
    lineup := 1
  end {if lineup}
end; {for k}

writeln (nsa, 'GE 0');
writeln (nsa)
end; {for j}
end {for i}
end; {if choice}

```

{ B&B side constraint, setting flow to 1 for arcs, is needed }

```

if bbset then {Branch & Bound user inputs are required}
begin
  for i := 1 to 50 do for j := 1 to 8 do bbarc[i,j] := ' ';
  writeln;
  writeln ('*****');
  writeln ('NOTE: Be VERY careful when you type in the arc name(s).');
  writeln ('-- Type the first arc name; press [RETURN]');
  writeln ('-- Type the second arc name; press [RETURN]');
  writeln ('-- ... and so on for the', setarcs:3, ' arc(s).');
  writeln ('SECOND NOTE: arcs to the destination cannot be set. ');
  writeln ('*****');
  writeln;
  for i := 1 to setarcs do readln (bbarc[i]); {read in arc name(s)}

```

```

arcch := ' ';
k := 0; {infinite loop preventer}
while ( (arcch = ' ') and (k<10) )do {move spaces in name (if any) to rear}
begin {in other words, left justify the arc name}
  k := k+1;
  for i := 1 to setarcs do
  begin
    arcch := bbarc[i,1];
    if arcch = ' ' then
    begin
      for j := 1 to 7 do bbarc[i,j] := bbarc[i,j+1];
      bbarc[i,8] := arcch
    end {if arcch}
  end {for i}
end; {while arcch}

```

{ Note: the file arcname will be searched until a match is made with the bbarc(s). }  
 { there will be coefficients of 0 for all arcs except the B&B arc(s) whose coef = 1 }

```

nomatch := 0; { used to keep track of no matches }
for i := 1 to setarcs do
begin
  bbset := true;
  counter[i] := 0; {will indicate which arc needs to be set to 1}
  reset (arcname);
  while ( (not eof(arcname)) and (bbset) ) do {until a match is found...}
  begin
    counter[i] := counter[i]+1;
    readln (arcname, bbcopy[i]);
    if (bbcopy[i] = bbarc[i]) then bbset := false
  end; {while not eof ...}
  if (bbset) then {if the arc wasn't found ... }
  begin
    counter[i] := 0; { reset so that the last arc is not flagged }
    nomatch := nomatch+1;
    writeln;
    writeln ('*****');
    writeln;
    writeln ('Arcname ', bbarc[i], ' was not found. ');
    writeln ('You may re-run the "areawall" program if you still');
    writeln ('need to set the flow on the arcs. ')
  end {if bbset}
end; {for i}

```

{ write the necessary B&B side constraint to set flow to 1 for the given arcs }  
 { if, as we count thru the arcs, we match the B&B arc's place, write a 1 }

```

lineup := 1;
for i := 1 to totarcs do
begin
  match := false;

```

```

{ the comparison below has to be done this way since there is no guarantee that }
{ the arcs whose flows are to be set to 1 are sequentially stored in "counter[]" }
    for j := 1 to setarcs do
        if i=counter[j] then match := true;      { we have a match in the list }
        if match then write (nsa, '1 ')
            else write (nsa, '. ');

        lineup := lineup+1;
        if lineup>40 then
            begin
                writeln (nsa);
                lineup := 1
            end {if lineup}
        end; {for i}

        writeln (nsa, 'EQ', (setarcs-nomatch):3);
        writeln (nsa)
    end; {if bbset}

    writeln (nsa, ',')
end; {if sidecon}

```

```

{ write code for executing PROC NETFLOW }

```

```

writeln (nsa, 'proc netflow');
writeln (nsa, '  nodedata=noded');
writeln (nsa, '  arcdata=arcd');
if sidecon then
    begin
        writeln (nsa, '  condata=cond');
        writeln (nsa, '  conout=solution;')
    end {if sidecon then}
else writeln (nsa, '  arcout=solution;');
writeln (nsa, 'run;');
writeln (nsa, 'print arcs/nonzero; run;');

```

```

{ *****

```

```

Instruct the user to run the sas program. }

```

```

writeln;
writeln;
writeln ('*****');
writeln;
writeln ('The SAS input file has been generated.');
```

```

writeln;
writeln ('1: Type "runsas nsa.dat" (no quotes) at the prompt; press [RETURN].');
writeln ('  When prompted for "VMS data file", press [RETURN] again.');
```

```

writeln ('  When prompted for the "queue", press [RETURN].');
writeln ('  ----- Note: if the 2-minute CPU time limit is reached,');
writeln ('              pick the long queue on the next run -----');
```

```

writeln;
writeln ('2: WAIT for the SAS program to solve the network, at which time');
writeln ('    you will receive the following:');
writeln ('        "Job RUNSAS (queue SSAS$QUEUE, entry ###) completed"');
writeln;
writeln ('3: Type "type nsa.log" (without quotes) to view the solution');
writeln ('        and information about the run. ');
writeln ('    Type "type nsa.lis" (without quotes) to view the arc flows');
writeln ('        (only arcs with nonzero flows will be shown)');
writeln;
writeln ('4: If the solution is not integer for all arcs (split flows, ');
writeln ('    - write down all the arcs that have flow on them; ');
writeln ('    - rerun the "areawall" program, this time choosing the ');
writeln ('        situation option #2 (meaning you want to begin a B&B process. ');
writeln ('    - you will be asked for the arc name(s) on which to set flow. ');
writeln;
end. {areawall}

```

### C.3 The "CELLBORDER" Pascal Program

```
program cellborder(input, output, rcddata, optdata, arcname, arcddata,
                  mdata, bedata, bndata, cdata, didata, working, nsa);
{
This is the final network implementation of subregion allocation.
}
{
Nodes are used to represent cells, cell borders, and connections between cells.
  Arcs flow from the source to a matrix wall, to a border wall, to a
  connect wall, and from the border and connect walls to any of five destinations
  depending on the options the user picks when running the "cellborder" program.
  One destination demands a flow equal to the number of borders comprising the
  subregion perimeter; the second, the subregion internal cell row connections;
  the third, the subregion internal cell column connections; the fourth, the sum of
  the internal row and column connections; and the fifth, the non-allocated cell borders.
  Flow of 4*n is supplied at the source (NODE S). Arcs of capacity 4 (no cost)
  go from S to the matrix wall cells (WALL M), one for each cell.
  Arcs of capacity 1 go from the wall M to the border wall (WALL B),
  one for each border node. Each border node in turn has one arc of capacity 1
  which can carry flow to the connect wall (WALL C), or to one of the
  external destination nodes (NODE DE or NODE DN). Only those border nodes
  which are not on the matrix perimeter can be connected to wall C.
  The connect nodes of wall C then have one arc each of capacity 2 which
  flows into the internal row connections destination node (NODE DIR),
  the internal column connections destination node (NODE DIC), or to the internal
  connections destination node (NODE DI).
  Each border node in wall B represents a cell border; each connect node in wall C,
  a possible internal cell connection.
  Every border of every cell is accounted for.
}
{
The row/column format for representing the working data matrix is
  used for the calculations. A return to the sequentially numbered matrix cells
  is used for node and arc names since they cannot exceed 8 characters in length.
}
{
The _space variables are used to determine data character length for file writing.
}
{
The following is an explanation of the files and what they are used for:
  rcddata: holds the values of the "rows" and "columns" variables of the matrix;
  optdata: holds the values of the user's options, choices, and subregion area;
  arcname: holds the list of arcnames;
  arcddata: holds values indicating whether arcs are internal or external row
            or column arcs;
  mdata: holds values on the wall M nodes in correct S-to-wall-M placement;
  bedata: holds values on the wall B nodes indicating their wall M parent
            in correct M-to-wall-B placement;
  bndata: holds values on the wall B nodes indicating their wall M parent
            in correct B-to-DN placement;
```

```

cdata: holds values on the wall C nodes indicating their wall B parents
      in correct B-to-wall-C placement;
didata: holds values on the wall C nodes indicating their place in the
      arc list (arcs to the DIR, DIC, or DI destinations);
working: holds the working file cost matrix;
nsa: the SAS input file which this program ultimately creates.
Note: rcdata, optdata, and working were created by the subnets program.
Note: arcname, arcdata, mdata, bedata, bndata, and cdata are used in
      the writing of side constraints.
}
type string = packed array [1..50, 1..8] of char;

var
  working: file of integer;
  rcdata, optdata, arcname, arcdata, mdata, bedata, bndata,
  cdata, didata, nsa: text;
  bbarc: string;                                {allows several arcs to have flows set}
  bbcopy: string;                                {used for determining position of arc names}
  i, j, k, lineup, exclude, per, introw, intcol, inttot, mnum, bnum, cnum, cellnum,
  mspace, bspace, cspace, rmicell, rmispace, rpicell, rpispace, cmicell,
  cmispace, cpicell, cpspace,
  cellcopy, rows, columns, subs, height, width,
  size, area, gmin, gmax, option, choice,
  totarcs, setarcs, nomatch, arctype, ccount: integer;
  selection: array [1..10] of integer;           {for storing all the user's SAS options}
  counter: array [1..50] of integer;             {for finding arc names in the arcname file}
  flow: array [1..100] of integer;               {for setting arc flows in B&B portion}
  cell: array [1..100, 1..100] of integer;
  arcch: char;
  sidecon, needit, compact, bbset, match: boolean; {for constraints and options}
                                          {bbset indicates whether the user wants to set flow in an arc}

begin {cellborder}
  reset (rcdata);
  reset (optdata);
  reset (working);

  read (optdata, option);
  read (optdata, choice);
  read (optdata, subs);
  read (optdata, area);
  sidecon := false;
  needit := false;
  compact := true;    {the default compact setting will be used unless user changes}
  bbset := false;
  read (rcdata, rows);
  read (rcdata, columns);
  size := (rows)*(columns);
{
*****

```

Determine the most compact subregion of the user's specified area. The variables *i* and *j* are used to represent rows and columns, but in the end, no distinction (restriction) is made on the orientation of the subregion, only on its shape in terms of perimeter borders and internal borders. Of course, the user may always elect to specify shape constraints when he chooses his options from the options list below.

Letting *i* and *j* represent rows and columns, sequentially build up an artificial subregion by alternately increasing *i* and *j* until its area equals or exceeds the user's specified area. If the artificial area exceeds, then drop it back down.

```

}
  i:= 1;
  j:= 1;
  while (area>(i*j)) do if (i>j) then j:= j+1 else i:= i+1;
  if ((i*j)>area) then
    begin
      if (i>j) then i:= i-1 else j:= j-1;
      needit:= true
    end; {if (i*j)>area}
}

```

Determine the most compact border allocation scheme based on the determined rectangular shape of the subregion.

```

}
  per:= 2*(i+j);
  inttot := (2*(j-1)*(i)) + (2*(i-1)*(j));
  exclude := (4*size)-(4*area);
{
  If at first the artificial subregion had exceeded the user's
  specified area, then, after decreasing the artificial subregion,
  there are still left-over cells which need to be connected.
  The left-over cells will all fit on one row (or column) since either
  i or j was previously decreased by only 1.
}

```

```

  if needit then
    begin
      per:= per+2;
      inttot:= inttot + (2*((2*(area-(i*j))) - 1));
      needit:= false
    end; {if needit}
}

```

```

*****
}

```

```

  for i := 1 to rows+2 do
    begin
      for j := 1 to columns+2 do
        begin
          read (working, cell[i,j])
        end
      end;
    end;
  {read in working file}

```



```

{
*****

User options determined.
Variable selection[1] is used to control the looping.
}

for i:= 1 to 10 do selection[i] := 0;
i:= 1;
while (selection[i]<>9) do
begin
  for i:= 1 to 10 do selection[i] := 0;
  i:= 1;
  writeln;
  writeln ('*****: *****');
  writeln ('Be sure you fully understand the following directions;');
  writeln ('your problem will NOT be solved correctly if you fail to');
  writeln ('properly enter in your options.');
```

- writeln ('\*\*\*\*\*');
- writeln ('DIRECTIONS FOR SELECTING AND ENTERING OPTIONS:');
- writeln ('1) Decide which options you wish to employ;');
- writeln ('2) Enter in the number(s) corresponding to your option(s);');
- writeln (' and press [RETURN] -- FOR EACH OPTION;');
- writeln (' (DO NOT enter in more than one option at a time');
- writeln (' or more than one option number on a line)');
- writeln ('3) After you have entered in your last option ...');
- writeln (' -- enter in the option number "9" to quit;');
- writeln ('4) NOTE: unless you are experimenting with constraints,');
- writeln (' you MUST enter in at least the following options:');
- writeln (' Version 1 requires options 1 and 2');
- writeln (' Version 2 requires option 5');
- writeln ('5) NOTE: entering in option selection "8" BEFORE QUITTING');
- writeln (' will cause the options list to re-display and will allow');
- writeln (' you to re-select your options.');
- writeln ('\*\*\*\*\*');
- writeln ('The following is your options list:');
- writeln;
- writeln ('1: Set flow constraints on wall-B-to-wall-C arcs.');
- writeln (' (two borders for each connector must contribute equally)');
- writeln ('2: Set flow constraints on wall-M-to-wall-B arcs.');
- writeln (' (every border of an allocated cell must contribute equally)');
- writeln (' (Not needed with option 5, below.));
- writeln ('3: Set compactness constraints on S-to-wall-M arcs.');
- writeln (' (minimum of two 1st-order neighbors for each cell)');
- writeln (' (Infeasible for some subregions of odd area.));
- writeln (' (Not needed with option 5, below.));
- writeln ('4: Set flow constraints relating S-to-M arcs with B-to-C arcs.');
- writeln (' (sum of C-arcs of a cell must be GE 1/2 the flow into the cell)');
- writeln (' (Infeasible for some subregions of odd area.));
- writeln (' (Infeasible with option 5, below.));
- writeln ('5: Create the network so that ALL matrix cells are accounted for.');
- writeln (' (the 4th destination node (DN), wall-B-to-DN arcs,');

```

        writeln ('      adjusted source flows, and B-to-DN contribution constraints'))';
writeln ('6: Set subregion shape constraints.');
```

writeln (' (you want a specified rectangular shape for your subregion)');

```

writeln ('7: Set the flow on one or more of the arcs.');
```

writeln (' (force a flow quantity on a previously split-flow solution)');

```

writeln ('8: Repeat the instructions and the options list.');
```

writeln (' (you made a mistake and wish to start over)');

```

writeln ('9: DONE.');
```

writeln (' (you have entered in all the options you want)');

```

writeln ('*****');
while ( (selection[i]<8) and (i<10) ) do
begin
    i:= i+1;
    readln (selection[i])
end {while selection<8 ...}
end; {while selection<>9}
```

for i:= 1 to 10 do if  
 ( (selection[i]<6) or (selection[i]=7) ) then sidecon := true;  
{side constraints are needed}

{

\*\*\*\*\*

Total network arcs depends on the network formulation chosen

total network arcs = the number of arcs from S to wall M = (size)  
 plus arcs from wall M to wall B = 1 for each cell border = (4\*size)  
 plus arcs from wall B to wall C = total matrix neighbor count =  
 2 for each corner cell, 3 for each non-corner perimeter cell, and  
 4 for all internal cells =  $(2*4) + (3*2*(rows-2)) + (3*2*(columns-2))$   
 +  $4*(rows-2)*(columns-2)$  =  
 =  $(8 + 6*[(rows-2)+(columns-2)] + 4*(rows-2)*(columns-2))$  =  
 = every cell border arc except those on the matrix perimeter =  
=  $(4*size) - (2*rows) - (2*columns)$   
 plus arcs to the destinations DE, (DIR and DIC), or DI.  
 = the number of arcs to wall B plus 1/2 those to wall C  
=  $(4*size) + (2*size) - rows - column$   
 therefore, the total number of network arcs is:  
 $(15*size) - 3*rows - 3*columns$ .

If the alternate, destination node DN, formulation was chosen, then the  
 total number of network arcs is increased by 4\*size (border nodes to DN).

}

```

totarcs := (15*size) - (3*rows) - (3*columns);
for i:= 1 to 10 do if (selection[i]=5) then needit := true;
if needit then totarcs := totarcs + (4*size);
needit := false;
```

{

\*\*\*\*\*

}

```

for i:= 1 to 10 do if (selection[i]=6) then needit := true;
if needit then
begin
```

```

compact:= false;           {the default compact setting will not be used}
height := 0;
width := 0;
while ( (height<1) or (height>rows) or (width<1) or (width>columns)
or ((height*width)<>area) ) do
begin
  writeln;
  writeln ('*****');
  writeln ('Your subregion size cannot exceed your matrix size;');
  writeln ('Your subregion height and width must equal');
  writeln (' your subregion area. ');
  writeln;
  writeln ('Enter in your subregion height. ');
  readln (height);
  writeln;
  writeln ('Enter in your subregion width. ');
  readln (width)
end; {while height ... }
per:= 2*(height+width);           {set subregion border characteristics}
introw := 2*(width-1)*(height);   {based on user inputs}
intcol := 2*(height-1)*(width);
exclude := (4*size)-(4*area);
needit:= false
end; {if needit}

{
*****

B&B side constraint, setting flow for arcs, if needed.
}

for i:= 1 to 10 do if (selection[i]=7) then needit := true;
if needit then           {B&B user inputs are required}
begin
  setarcs := 0;
  for i := 1 to 100 do flow[i] := 0;
  bbset := true;
  writeln;
  writeln ('*****');
  writeln ('How many arcs need to have their flow set? ');
  while ( (setarcs<1) or (setarcs>50) ) do
  begin
    writeln ('(the number must be greater than 0)');
    writeln (' and should not exceed 50). ');
    writeln;
    readln (setarcs)
  end; {while setarcs}

  for i := 1 to 50 do for j := 1 to 8 do bbarc[i,j] := ' ';
  writeln;
  writeln ('***** ENTER IN YOUR ARC INFORMATION *****');
  writeln (' ***** BE VERY CAREFUL !! ***** ');
  writeln ('-- Type the first arc name and press [RETURN] ');

```

```

writeln ('----then type the flow you want on that arc and press [RETURN]');
writeln ('-- Type the second arc name (if any) and press [RETURN]');
writeln ('----then type the flow you want on the second arc and press [RETURN]');
writeln ('-- ... and so on for the', setarcs:3, ' arc(s).');
writeln ('*****');
writeln;
for i := 1 to setarcs do
begin
    readln (bbarc[i]);
    readln (flow[i])
end; {for i}

arcch := ' ';
k := 0;
while ( arcch = ' ') and (k<10) do
begin
    k := k+1;
    for i := 1 to setarcs do
    begin
        arcch := bbarc[i,1];
        if arcch = ' ' then
        begin
            for j := 1 to 7 do bbarc[i,j] := bbarc[i,j+1];
            bbarc[i,8] := arcch
        end {if arcch}
    end {for i}
end; {while arcch}
needit:= false
end; {if needit}

{
*****
Create SAS file NSA.DAT composed of node info, arc info, and side constraints.
Frame cells will not be in the network.
Periods represent coefficients of 0.
}

rewrite (nsa);
rewrite (arcname);
rewrite (arcdata);
rewrite (mdata);
rewrite (bedata);
rewrite (bndata);
rewrite (cdata);
rewrite (didata);

writeln (nsa, 'data noded:');
writeln (nsa, '    input _node_ $ _sd_');
writeln (nsa, '    cards');
for i:= 1 to 10 do if (selection[i]=5) then needit:= true;
if needit then area:=size;
needit:= false;

```

```

writeln (nsa, 'S ', 4*area:5);
writeln (nsa, 'DE ', (0-per):5);
if compact
then writeln (nsa, 'DI ', (0-inttot):5)
else begin
  writeln (nsa, 'DIR ', (0-introw):5);
  writeln (nsa, 'DIC ', (0-intcol):5)
end; {if compact, else}
for i:= 1 to 10 do if (selection[i]=5) then needit := true;
if needit then writeln (nsa, 'DN ', (0-exclude):5);
needit := false;
writeln (nsa, ','');
writeln (nsa, 'data arcd;');
writeln (nsa, 'input _from_$ _to_$ _cost_ _capac_ _name_$;');
writeln (nsa, 'cards;');
{
*****

```

Write to the data files the arc info from S to wall M.  
 Capacities on these arcs are 4 each, 1 unit for each border node.  
 }

```

  mspace := 1;
  for i := 2 to rows+1 do
  begin
    for j := 2 to columns+1 do
    begin

      mnum := (((i-1)*(columns+2))+j);
      if mnum > 9 then mspace := 2;
      if mnum > 99 then mspace := 3;

      write (nsa, 'S ', 'm',mnum:mspace);
      writeln (nsa, ' . 4 Sm',mnum:mspace);
      writeln (arcname, 'Sm',mnum:mspace);
      writeln (arcdata, '8888');
      writeln (mdata, mnum);
      writeln (bedata, '8888');
      writeln (bndata, '8888');
      writeln (cdata, '8888');
      writeln (didata, '8888');

      {determine current cell}
      {adjust character width if need be}
      {neither a row nor column arc}
      {used for compactness constraints}
      {cannot be used for contiguity constraints}
      {cannot be used for contiguity constraints}
      {cannot be used for contiguity constraints}
      {not a DI-type arc}

    end {for j}
  end; {for i}
{
*****

```

Write the arc info from wall M to wall B, from wall B to wall C  
 and from walls B and C to their respective destinations:  
 The B-wall nodes are sequentially numbered clockwise, around each cell,  
 4 for each M-wall cell.  
 Capacities on the M-to-wall-B arcs are limited to 1.

The C-wall nodes are numbered and ordered the same way if they exist.

Capacities on the B-to-wall-C arcs are limited to 1.

The cost checking of neighbor cells for frame cells determines if a connector node exists.

Arc types are based on whether the arcs carry flow within a row or within a column and are:

1=neighbor row; 2=neighbor col; 3=internal row; 4=internal col.

```
}
  mspace := 1;                                {cell character widths}
  rmispace := 1;
  cpispace := 1;
  rpispace := 1;
  cmispace := 1;
  ccount := 0;                                {keeps tally of internal connector nodes}

  for i := 2 to rows+1 do
  begin
    for j := 2 to columns+1 do
    begin

      mnum := ( ((i-1)*(columns+2))+j );        {determine current cell and nodes}
      bnum := ( ((i-2)*(columns)) + (j-1) );
      rmicell := ((bnum-1)*4) + 1;
      cpicell := ((bnum-1)*4) + 2;
      rpicell := ((bnum-1)*4) + 3;
      cmicell := ((bnum-1)*4) + 4;

      if mnum > 9 then mspace := 2;              {adjust character widths if need be}
      if mnum > 99 then mspace := 3;
      if rmicell > 9 then rmispace := 2;
      if rmicell > 99 then rmispace := 3;
      if cpicell > 9 then cpispace := 2;
      if cpicell > 99 then cpispace := 3;
      if rpicell > 9 then rpispace := 2;
      if rpicell > 99 then rpispace := 3;
      if cmicell > 9 then cmispace := 2;
      if cmicell > 99 then cmispace := 3;

    {
  Write the arc info to files NSA, arcname, arcdata, mdata, bedata, bndata, cdata.

  Wall M to wall B.
  }

  write (nsa, 'm',mnum:mspace,' ','b',rmicell:rmispace);
  write (nsa, ' . 1 ');
  writeln (nsa, 'm',mnum:mspace,'b',rmicell:rmispace);
  writeln (arcname, 'm',mnum:mspace,'b',rmicell:rmispace);
  writeln (arcdata, '2');                                {neighbor column arc}
  writeln (mdata, '8888');
  writeln (bedata, bnum);
  writeln (bndata, '8888');
```

```

        writeln (cdata, '8888');
        writeln (didata, '8888');                                {not a DI-type arc}
    {
        Wall B to DE.
    }

    write (nsa, 'b',rmicell:rmispace,' DE', cell[i,j]:4, ' 1 ');
    writeln (nsa, 'b',rmicell:rmispace,'DE');
    writeln (arcname, 'b',rmicell:rmispace,'DE');
    writeln (arcddata, '8888');                                    {not any kind of row or column arc}
    writeln (mdata, '8888');
    writeln (bedata, '8888');                                    {cannot be used for contiguity constraints}
    writeln (bndata, '8888');
    writeln (cdata, '8888');
    writeln (didata, '8888');                                    {not a DI-type arc}
    {
        Wall B to DN (if needed).
    }

    for k:= 1 to 10 do if (selection[k]=5) then needit := true;
    if needit then
    begin
        write (nsa, 'b',rmicell:rmispace,' DN . 1 ');
        writeln (nsa, 'b',rmicell:rmispace,'DN');
        writeln (arcname, 'b',rmicell:rmispace,'DN');
        writeln (arcddata, '8888');                                {not any kind of row or column arc}
        writeln (mdata, '8888');
        writeln (bedata, '8888');                                    {cannot be used for contiguity constraints}
        writeln (bndata, bnum);
        writeln (cdata, '8888');
        writeln (didata, '8888');                                    {not a DI-type arc}
        needit:= false
    end; {if needit}

    {
        Wall B to Wall C and Wall C to DI/DIR/DIC, if required.
    }

    if (cell[i-1,j]<>888) then {the rpi connector of the above cell is used}
    begin
        cspace:= 1;
        cnum := ( (i-3)*(2*(columns-1)+1) + (2*(j-2)) + 2);
        if j=(columns+1) then cnum := cnum-1;
        if cnum > 9 then cspace := 2;
        if cnum > 99 then cspace := 3;
        write (nsa, 'b',rmicell:rmispace,' ', 'c',cnum:cspace);
        write (nsa, cell[i,j]:4, ' 1 ');
        writeln (nsa, 'b',rmicell:rmispace,'c',cnum:cspace);
        writeln (arcname, 'b',rmicell:rmispace,'c',cnum:cspace);
        writeln (arcddata, '4');                                    {internal column arc}
        writeln (mdata, '8888');
        writeln (bedata, '8888');                                    {cannot be used for contiguity constraints}
        writeln (bndata, '8888');
        writeln (cdata, cnum);                                    {used for contiguity}
        writeln (didata, '8888')                                    {not a DI-type arc}
    end;

```

```

        end; {if cell[i-1,j]}
    {
    Wall M to wall B.
    }

    write (nsa, 'm',mnum:mspace,' ','b',cpicell:cpispace);
    write (nsa, ' . 1 ');
    writeln (nsa, 'm',mnum:mspace,'b',cpicell:cpispace);
    writeln (arcname, 'm',mnum:mspace,'b',cpicell:cpispace);
    writeln (arcddata, '1'); {neighbor row arc}
    writeln (mdata, '8888');
    writeln (bedata, bnum);
    writeln (bndata, '8888');
    writeln (cdata, '8888');
    writeln (didata, '8888'); {not a DI-type arc}

    {
    Wall B to DE.
    }

    write (nsa, 'b',cpicell:cpispace,' DE', cell[i,j]:4, ' 1 ');
    writeln (nsa, 'b',cpicell:cpispace,'DE');
    writeln (arcname, 'b',cpicell:rmispace,'DE');
    writeln (arcddata, '8888'); {not any kind of row or column arc}
    writeln (mdata, '8888');
    writeln (bedata, '8888'); {cannot be used for contiguity constraints}
    writeln (bndata, '8888');
    writeln (cdata, '8888');
    writeln (didata, '8888'); {not a DI-type arc}

    {
    Wall B to DN (if needed).
    }

    for k:= 1 to 10 do if (selection[k]=5) then needit := true;
    if needit then
    begin
        write (nsa, 'b',cpicell:cpispace,' DN . 1 ');
        writeln (nsa, 'b',cpicell:cpispace,'DN');
        writeln (arcname, 'b',cpicell:cpispace,'DN');
        writeln (arcddata, '8888'); {not any kind of row or column arc}
        writeln (mdata, '8888');
        writeln (bedata, '8888'); {cannot be used for contiguity constraints}
        writeln (bndata, bnum);
        writeln (cdata, '8888');
        writeln (didata, '8888'); {not a DI-type arc}
        needit := false;
    end; {if needit}

    {
    Wall B to Wall C and Wall C to DI/DIR/DIC, if required.
    }

    if (cell[i,j+1]<>888) then {a connector is needed}
    begin
        ccount := ccount+1;
        cspace:= 1;
        cnum := ( (i-2)*(2*(columns-1)+1) + (2*(j-2)) + 1);
    end;

```



```

if i=(rows+1) then cnum := cnum-(j-2);
if cnum > 9 then cspace := 2;
if cnum > 99 then cspace := 3;
write (nsa, 'b',cpicell:cpispace,' ','c',cnum:cspace);
write (nsa, cell[i,j]:4, ' 1 ');
writeln (nsa, 'b',cpicell:cpispace,'c',cnum:cspace);
writeln (arcname, 'b',cpicell:cpispace,'c',cnum:cspace);
writeln (arcdata, '3'); {internal row arc}
writeln (mdata, '8888');
writeln (bedata, '8888'); {cannot be used for contiguity constraints}
writeln (bndata, '8888');
writeln (cdata, cnum); {used for contiguity}
writeln (didata, '8888'); {not a DI-type arc}
if compact
then begin
write (nsa, 'c',cnum:cspace,' DI . 2 ');
writeln (nsa, 'c',cnum:cspace,'DI');
writeln (arcname, 'c',cnum:cspace,'DI')
end {if compact, then}
else begin
write (nsa, 'c',cnum:cspace,' DIR . 2 ');
writeln (nsa, 'c',cnum:cspace,'DIR');
writeln (arcname, 'c',cnum:cspace,'DIR')
end; {if compact, else}
writeln (arcdata, '8888'); {not any kind of row or column arc}
writeln (mdata, '8888');
writeln (bedata, '8888'); {cannot be used for contiguity constraints}
writeln (bndata, '8888');
writeln (cdata, '8888');
writeln (didata, cnum) {not a DI-type arc}
end; {if cell[i,j+1]}

{
Wall M to wall B.
}

write (nsa, 'm',mnum:mspace,' ','b',rpicell:rpispace);
write (nsa, ' . 1 ');
writeln (nsa, 'm',mnum:mspace,'b',rpicell:rpispace);
writeln (arcname, 'm',mnum:mspace,'b',rpicell:rpispace);

writeln (arcdata, '2'); {neighbor column arc}
writeln (mdata, '8888');
writeln (bedata, bnum);
writeln (bndata, '8888');
writeln (cdata, '8888');
writeln (didata, '8888'); {not a DI-type arc}

{
Wall B to DE.
}

write (nsa, 't',rpicell:rpispace,' DE', cell[i,j]:4, ' 1 ');
writeln (nsa, 'b',rpicell:rpispace,'DE');
writeln (arcname, 'b',rpicell:rpispace,'DE');

```

```

        writeln (arcddata, '8888');           {not any kind of row or column arc}
        writeln (mdata, '8888');
        writeln (bedata, '8888');           {cannot be used for contiguity constraints}
        writeln (bndata, '8888');
        writeln (cdata, '8888');
        writeln (didata, '8888');           {not a DI-type arc}
    {
    Wall B to DN (if needed).
    }

    for k:= 1 to 10 do if (selection[k]=5) then needit := true;
    if needit then
    begin
        write (nsa, 'b',rp1cell:rp1space,' DN . 1 ');
        writeln (nsa, 'b',rp1cell:rp1space,'DN');
        writeln (arcname, 'b',rp1cell:rp1space,'DN');
        writeln (arcddata, '8888');           {not any kind of row or column arc}
        writeln (mdata, '8888');
        writeln (bedata, '8888');           {cannot be used for contiguity constraints}
        writeln (bndata, bnum);
        writeln (cdata, '8888');
        writeln (didata, '8888');           {not a DI-type arc}
        needit := false
    end; {if needit}

    {
    Wall B to Wall C and Wall C to DI/DIR/DIC, if required.
    }

    if (cell[i+1,j]<>888) then           {a connector is needed}
    begin
        ccount := ccount+1;
        cspace:= 1;
        cnum := ( (i-2)*(2*(columns-1)+1) + (2*(j-2)) + 2);
        if j=(columns+1) then cnum := cnum-1;
        if cnum > 9 then cspace := 2;
        if cnum > 99 then cspace := 3;
        write (nsa, 'b',rp1cell:rp1space,' ', 'c',cnum:cspace);
        write (nsa, cell[i,j]:4, ' 1 ');
        writeln (nsa, 'b',rp1cell:rp1space,'c',cnum:cspace);
        writeln (arcname, 'b',rp1cell:rp1space,'c',cnum:cspace);
        writeln (arcddata, '4');           {internal column arc}
        writeln (mdata, '8888');
        writeln (bedata, '8888');           {cannot be used for contiguity constraints}
        writeln (bndata, '8888');
        writeln (cdata, cnum);           {used for contiguity}
        writeln (didata, '8888');           {not a DI-type arc}
        if compact
        then begin
            write (nsa, 'c',cnum:cspace,' DI . 2 ');
            writeln (nsa, 'c',cnum:cspace,'DI');
            writeln (arcname, 'c',cnum:cspace,'DI')
        end {if compact, then}
        else begin

```

```

        write (nsa, 'c', cnum: cspace, ' DIC . 2 ');
        writeln (nsa, 'c', cnum: cspace, 'DIC');
        writeln (arcname, 'c', cnum: cspace, 'DIC')
    end; {if compact; else}
    writeln (arcdata, '8888');           {not any kind of row or column arc}
    writeln (mdata, '8888');
    writeln (bedata, '8888');           {cannot be used for contiguity constraints}
    writeln (bndata, '8888');
    writeln (cdata, '8888');
    writeln (didata, cnum)               {not a DI-type arc}
end; {if cell[i+1,j]}

{
Wall M to wall B.
}

    write (nsa, 'm', mnum: mspace, ' ', 'b', cmicell: cmispace);
    write (nsa, ' ' . 1 ');
    writeln (nsa, 'm', mnum: mspace, 'b', cmicell: cmispace);
    writeln (arcname, 'm', mnum: mspace, 'b', cmicell: cmispace);
    writeln (arcdata, '1');               {neighbor row arc}
    writeln (mdata, '8888');
    writeln (bedata, bnum);
    writeln (bndata, '8888');
    writeln (cdata, '8888');
    writeln (didata, '8888');             {not a DI-type arc}

{
Wall B to DE.
}

    write (nsa, 'b', cmicell: cmispace, ' DE', cell[i,j]:4, ' 1 ');
    writeln (nsa, 'b', cmicell: cmispace, 'DE');
    writeln (arcname, 'b', cmicell: cmispace, 'DE');
    writeln (arcdata, '8888');           {not any kind of row or column arc}
    writeln (mdata, '8888');
    writeln (bedata, '8888');           {cannot be used for contiguity constraints}
    writeln (bndata, '8888');
    writeln (cdata, '8888');
    writeln (didata, '8888');             {not a DI-type arc}

{
Wall B to DN (if needed).
}

    for k:= 1 to 10 do if (selection[k]=5) then needit := true;
    if needit then
    begin
        write (nsa, 'b', cmicell: cmispace, ' DN . 1 ');
        writeln (nsa, 'b', cmicell: cmispace, 'DN');
        writeln (arcname, 'b', cmicell: cmispace, 'DN');
        writeln (arcdata, '8888');           {not any kind of row or column arc}
        writeln (mdata, '8888');
        writeln (bedata, '8888');           {cannot be used for contiguity constraints}
        writeln (bndata, bnum);
        writeln (cdata, '8888');
        writeln (didata, '8888');             {not a DI-type arc}
    end
end

```

```

        needit := false
    end; {if needit}
}
Wall B to Wall C and Wall C to DI/DIR/DIC, if required.
}

    if (cell[i,j-1]<>888) then          {the cp1 connector of the left cell is used}
    begin
        cspace:= 1;
        cnum := ( (i-2)*(2*(columns-1)+1) + (2*(j-3)) + 1);
        if i=(rows+1) then cnum := cnum-(j-3);
        if cnum > 9 then cspace := 2;
        if cnum > 99 then cspace := 3;
        write (nsa, 'b',cmicell:cmispace,' ','c',cnum:cspace);
        write (nsa, cell[i,j]:4, ' 1 ');
        writeln (nsa, 'b',cmicell:cmispace,'c',cnum:cspace);
        writeln (arcname, 'b',cmicell:cmispace,'c',cnum:cspace);
        writeln (arcdata, '3');                      {internal row arc}
        writeln (mdata, '8888');
        writeln (bedata, '8888');                    {cannot be used for contiguity constraints}
        writeln (bndata, '8888');
        writeln (cdata, cnum);                        {used for contiguity}
        writeln (didata, '8888')                      {not a DI-type arc}
    end {if cell[i,j-1]}
    end {for j}
end; {for i}

    writeln (nsa, ';');
}
*****

SIDE CONSTRAINTS SECTION

All arcs will be listed since different constraints may utilize different arcs.

*****
}
    if sidecon then                                {side constraints of some kind are needed}
    begin

        reset (arcname);

        writeln (nsa, 'data cond;');
        writeln (nsa, 'input');
        j := 1;                                     {10 at a time}
        for i := 1 to totarcs do                    {list all arcs}
        begin
            bbcopy[1] := '                        '; {clear the variable}
            readln (arcname, bbcopy[1]);

            write (nsa, bbcopy[1], ' ');

```

```

j := j+1;
if j>10
then begin
    j := 1;
    writeln (nsa)
end {if j, then}
end; {for i}

writeln (nsa, '_type_$ _rhs_');
writeln (nsa, 'cards;');

{
*****
If arc-setting is required...
}

for i:= 1 to 10 do if (selection[i]=7) then needit := true;
if needit then {B&B user inputs are required}
begin
{
Note: the file arcname will be searched until a match is made with the bbarc(s).
There will be one additional side constraint for every arc flow that is set.
There will be coefficients of 0 for all arcs except the B&B arc(s) whose coef = 1.
}

nomatch := 0; { used to keep track of no matches }
for i := 1 to setarcs do
begin
    bbset := true;
    counter[i] := 0; {will indicate which arc needs to be set to 1}
    reset (arcname);
    while ( (not eof(arcname)) and (bbset) ) do {until a match is found...}
    begin
        counter[i] := counter[i]+1;
        readln (arcname, bbcopy[i]);
        if (bbcopy[i] = bbarc[i]) then bbset := false
    end; {while not eof ...}
    if (bbset) then {if the arc wasn't found ... }
    begin
        counter[i] := 0; { reset so that the last arc is not flagged }
        nomatch := nomatch+1;
        writeln;
        writeln ('*****');
        writeln;
        writeln ('Arcname ', bbarc[i], ' was not found. ');
        writeln ('You may re-run the "sasfile" program if you still');
        writeln ('need to set the flow on the arcs. ')
    end {if bbset}
end; {for i}

{
Write the necessary B&B side constraint to set the flow on the given arcs.
}

for i := 1 to setarcs do
begin

```

```

if counter[i]<>0          {if the arc was indeed found in the list ... }
then begin
  lineup := 1;
  for j := 1 to (counter[i]-1) do
  begin
    write (nsa, '. ');
    lineup := lineup+1;
    if lineup>40 then
    begin
      writeln (nsa);
      lineup := 1
    end {if lineup}
  end; {for j}
  write (nsa, '1 ');
  lineup := lineup+1;
  if lineup>40 then
  begin
    writeln (nsa);
    lineup := 1
  end; {if lineup}
  for j := 1 to (totarcs-counter[i]) do
  begin
    write (nsa, '. ');
    lineup := lineup+1;
    if lineup>40 then
    begin
      writeln (nsa);
      lineup := 1
    end {if lineup}
  end; {for j}
  writeln (nsa, 'EQ ', flow[i]:4);
  writeln (nsa)
end {if counter}
end; {for i}
needit:= false
end; {if needit}

```

```

writeln ('*****');
writeln;
writeln ('Creating the SAS input file ... please wait');
writeln;

```

```

{
*****

```

```

Write the contiguity constraints if needed
The first set establishes the fact that 2 adjacent borders must have 1 unit of
flow each go into their shared cnode, their connector node.
The cdata file contains the list of cnodes.
}

```

```

for i:= 1 to 10 do if (selection[i]=1) then needit := true;
if needit then

```

```

begin
  for i := 1 to ccount do           {once thru for each internal connector node}
  begin
    k := 1;                         {keeps track of which "1" coef gets the negative sign}
    reset (cdata);
    lineup:= 1;
    for j := 1 to totarcs do        {search thru entire cdata file}
    begin
      readln (cdata, cnum);         {get stored cnumber from file cdata}
      if ( (cnum = i) and (k = 1) )
      then begin
        write (nsa, '1 ');
        k := -k
      end {if cnum..., then}
      else begin
        if ( (cnum = i) and (k = -1) )
        then begin
          write (nsa, '-1 ');
          k := -k
        end {if cnum..., else; if cnum..., then}
        else write (nsa, '. ')
      end; {if cnum ..., else}
      lineup := lineup+1;
      if lineup>40 then
      begin
        writeln (nsa);
        lineup := 1
      end {if lineup}
    end; {for j}
    writeln (nsa, 'EQ 0');
    writeln (nsa)
  end; {for i}
  needit:= false
end; {if needit}

```

{

The second set establishes the fact that for any border of any cell to count as either a subregion external or internal border, all borders of that cell must count as one or the other; there must be a flow of 4 or 0 into each cell. The bedata file contains the list of bnums.

}

```

for i:= 1 to 10 do if (selection[i]=2) then needit := true;
if needit then
begin
  for i := 1 to (size) do           {once thru for each cell (each bnum) }
  begin
    k := 1;
    for ccount := 1 to 3 do         {3 constraints total for the 4 cell borders}
    begin
      reset (bedata);
      lineup:= 1;
      for j := 1 to totarcs do      {read thru entire list stored in file bedata}

```

```

begin
  readln (bedata, bnum);
  if (bnum = i) then
    begin
      case k of
        1 : write (nsa, '1 ');
        2 : write (nsa, '1 ');
        3 : write (nsa, '-1 ');
        4 : write (nsa, '-1 ');
        5 : write (nsa, '1 ');
        6 : write (nsa, '-1 ');
        7 : write (nsa, '1 ');
        8 : write (nsa, '-1 ');
        9 : write (nsa, '1 ');
        10 : write (nsa, '-1 ');
        11 : write (nsa, '-1 ');
        12 : write (nsa, '1 ');
      end; {case}
      k := k+1
    end {if bnum, then}
    else write (nsa, '. ');
    lineup := lineup+1;
    if lineup>40 then
      begin
        writeln (nsa);
        lineup := 1
      end {if lineup}
    end; {for j}
    writein (nsa, 'EQ 0');
    writeln (nsa)
  end {for ccount}
end; {for i}
needit:= false;
end; {if needit}
{
*****

Write the compactness constraints, if needed.

-2 is the coefficient for the source arc to the current cell
  and a 1 is the coefficient for the source arcs to the cell neighbors.
All other arcs will have coefficients of 0
The file mdata is searched for cell and neighbor matches
  to know where to properly place the coefficients .
}

for i:= 1 to 10 do if (selection[i]=3) then needit := true;
if needit then
begin
  for i := 2 to rows+1 do
  begin
    for j := 2 to columns+1 do
      {for each matrix cell ... }

```



```

begin
  reset (mdata);
  lineup := 1;
  mnum := (((i-1)*(columns+2))+j);           {determine the current cell}
  rmicell := (((i-2)*(columns+2))+j); {determine 1st-order neighbors ...}
  rpicell := ((i*(columns+2))+j);
  cmicell := (((i-1)*(columns+2))+(j-1));
  cpicell := (((i-1)*(columns+2))+(j+1));

  for k := 1 to totarcs do                     {search thru entire arcdata file}
  begin
    readln (mdata, cellcopy);                  {get stored mnum from file mdata}
    if cellcopy = mnum                         {if we are at the current cell number ...}
    then begin
      write (nsa, '-2 ');
      lineup := lineup+2
    end {if cellcopy, then}
    else begin                                  {else, if we are at a neighbor cell ... }
      if ((cellcopy=rmicell) or (cellcopy=rpicell) or (cellcopy=cmicell)
        or (cellcopy=cpicell))
      then write (nsa, '1 ')
      else write (nsa, '. ');                  {not the cell nor a neighbor}
      lineup := lineup+1
    end; {if cellcopy, else}
    if lineup>40 then
    begin
      writeln (nsa);
      lineup := 1
    end {if lineup}
  end; {for k}

  writeln (nsa, 'GE 0');
  writeln (nsa)
end {for j}
end; {for i}
needit:= false;
end; {if needit}
{

```

\*\*\*\*\*

Write the constraints relating cells to their B-to-C arcs, if needed.

1 is the coefficient for the wall-B-to-wall-C connector arcs,  
 and a -1 is the coefficient for the connector nodes' cell parent.  
 All other arcs will have coefficients of 0  
 The file mdata is searched for cell matches  
 to know where to properly place the cell coefficients.  
 The file didata is searched for B-to-C arc matches  
 to know where to properly place the connector arc coefficients.  
 The variables rplicell, rmicell, cpicell, and cmicell are used  
 to store the appropriate cnums (connector node numbers).

```

}
for i:= 1 to 10 do if (selection[i]=4) then needit := true;
if needit then
begin
  for i := 2 to rows+1 do {for each matrix cell ... }
  begin
    for j := 2 to columns+1 do
    begin
      mnum := (((i-1)*(columns+2))+j); {determine the current cell}
{
Determine all the cell's connector nodes.
}
      if (cell[i-1,j]<>888) then {the rpi connector of the above cell is used}
      begin
        rmicell := ( (i-3)*(2*(columns-1)+1) + (2*(j-2)) + 2);
        if j=(columns+1) then rmicell := rmicell-1
      end
      else rmicell := 0; {there is no rmi connector node}
      if (cell[i,j+1]<>888) then {a connector is needed}
      begin
        cpicell := ( (i-2)*(2*(columns-1)+1) + (2*(j-2)) + 1);
        if i=(rows+1) then cpicell := cpicell-(j-2);
      end
      else cpicell := 0; {there is no cpi connector node}
      if (cell[i+1,j]<>888) then {a connector is needed}
      begin
        rpicell := ( (i-2)*(2*(columns-1)+1) + (2*(j-2)) + 2);
        if j=(columns+1) then rpicell := rpicell-1;
      end
      else rpicell := 0; {there is no rpi connector node}
      if (cell[i,j-1]<>888) then {the cpi connector of the left cell is used}
      begin
        cmicell := ( (i-2)*(2*(columns-1)+1) + (2*(j-3)) + 1);
        if i=(rows+1) then cmicell := cmicell-(j-3);
      end
      else cmicell := 0; {there is no cmi connector node}

      reset (mdata); {prepare to read from the appropriate files}
      reset (didata);
      lineup := 1;

      for k := 1 to totarcs do {search thru entire arcdata file}
      begin
        readln (mdata, cellcopy); {get stored mnum from file mdata}
        readln (didata, cnum); {get stored cnum from file didata}
        if cellcopy = mnum {if we are at the current cell number ...}
        then begin
          write (nsa, '-1 ');
          lineup := lineup+2
        end {if cellcopy, then}
        else begin {else, if we are at a connector node ... }

```

```

        if ( (cnum= rmicell) or (cnum= cp1cell) or (cnum= rp1cell)
            or (cnum= cmicell) )
            then write (nsa, '1 ')
            else write (nsa, '. ');           {neither a source nor connector arc}
        lineup := lineup+1
    end; {if cellcopy, else}
    if lineup>40 then
    begin
        writeln (nsa);
        lineup := 1
    end {if lineup}
    end; {for k}

    writeln (nsa, 'GE 0');
    writeln (nsa)
    end {for j}
end; {for i}
needit:= false
end; {if needit}
{
*****

```

These side constraints are used with Version 2 of the network and state  
that for a border of any cell to count as a non-allocated subregion  
border, then all borders of that cell must count that way.  
The bndata file contains the list of bnums.  
}

```

for i:= 1 to 10 do if (selection[i]=5) then needit := true;
if needit then
begin
    for i := 1 to (size) do           {once thru for each cell (each bnum) }
    begin
        k := 1;
        for count := 1 to 3 do       {3 constraints total for the 4 cell borders}
        begin
            reset (bndata);
            lineup:= 1;
            for j := 1 to totarcs do  {read thru entire list stored in file bndata}
            begin
                readln (bndata, bnum);           {get stored bnumber from file bndata}
                if (bnum = i) then
                begin
                    case k of
                        1 : write (nsa, '1 ');
                        2 : write (nsa, '1 ');
                        3 : write (nsa, '-1 ');
                        4 : write (nsa, '-1 ');
                        5 : write (nsa, '1 ');
                        6 : write (nsa, '-1 ');
                        7 : write (nsa, '1 ');
                        8 : write (nsa, '-1 ');

```

```

        9 : write (nsa, '1 ');
        10 : write (nsa, '-1 ');
        11 : write (nsa, '-1 ');
        12 : write (nsa, '1 ');
    end; {case}
    k := k+1
end {if bnum, then}
else write (nsa, '. ');
lineup := lineup+1;
if lineup>40 then
begin
    writeln (nsa);
    lineup := 1
end {if lineup}
end; {for j}
writeln (nsa, 'EQ 0');
writeln (nsa)
end {for ccount}
end ;{for i}
needit:= false
end; {if needit}

    writeln (nsa, ';')
end; {if sidecon}
{
*****

Write code for executing PROC NETFLOW.
}

    writeln (nsa, 'proc netflow');
    writeln (nsa, '    nodedata=noded');
    writeln (nsa, '    arcdata=arcd');
    if sidecon then
    begin
        writeln (nsa, '    condata=cond');
        writeln (nsa, '    conout=solution;')
    end {if sidecon then}
    else writeln (nsa, '    arcout=solution;');
    writeln (nsa, 'run;');
    writeln (nsa, 'print arcs/nonzero; run;');
{
*****

Instruct the user to run the sas program.
}

    writeln;
    writeln;
    writeln ('*****');
    writeln;
    writeln ('The SAS input file has been generated.');
```

```

writeln ('1: Type "runsas nsa.dat" (no quotes) at the prompt; press [RETURN].');
writeln ('  When prompted for "VMS data file", press [RETURN] again.');
```

When prompted for the "queue", press [RETURN].');

```

writeln ('  ----- Note: if the 2-minute CPU time limit is reached,');
writeln ('           pick the long queue on the next run -----');
```

wait for the SAS program to solve the network, at which time you will receive the following:

```

writeln ('  "Job RUNSAS (queue SSAS$QUEUE, entry ###) completed"');
```

Type "type (or "more") nsa.log" (without quotes) to view the solution and information about the run.

```

writeln ('  Type "type (or "more") nsa.lis" (without quotes) to view the arc flows';
writeln ('           (only arcs with nonzero flows will be shown)');
```

end. {cellborder}

#### C.4 Input File for Problem AREAWALL1

```
data noded;
    input _node_$ _sd_;
    cards;
ss 1
d1 -1
;
data arcd;
input _from_$ _to_$ _cost_ _capac_ _name_$;
cards;
ss a8 6 1 ssa8
ss a9 5 1 sca9
ss a10 3 1 ssa10
ss a11 4 1 ssa11
ss a14 1 1 ssa14
ss a15 3 1 ssa15
ss a16 2 1 ssa16
ss a17 2 1 ssa17
ss a20 5 1 ssa20
ss a21 4 1 ssa21
ss a22 1 1 ssa22
ss a23 3 1 ssa23
ss a26 2 1 ssa26
ss a27 2 1 ssa27
ss a28 3 1 ssa28
ss a29 1 1 ssa29
a8 b9 5 1 a8b9
a8 b14 1 1 a8b14
a9 b10 3 1 a9b10
a9 b15 3 1 a9b15
a9 b8 6 1 a9b8
a10 b11 4 1 a10b11
a10 b16 2 1 a10b16
a10 b9 5 1 a10b9
a11 b17 2 1 a11b17
a11 b10 3 1 a11b10
a14 b8 6 1 a14b8
a14 b15 3 1 a14b15
a14 b20 5 1 a14b20
a15 b9 5 1 a15b9
a15 b16 2 1 a15b16
a15 b21 4 1 a15b21
a15 b14 1 1 a15b14
a16 b10 3 1 a16b10
a16 b17 2 1 a16b17
a16 b22 1 1 a16b22
a16 b15 3 1 a16b15
a17 b11 4 1 a17b11
a17 b23 3 1 a17b23
a17 b16 2 1 a17b16
```

a20 b14 1 1 a20b14  
a20 b21 4 1 a20b21  
a20 b26 2 1 a20b26  
a21 b15 3 1 a21b15  
a21 b22 1 1 a21b22  
a21 b27 2 1 a21b27  
a21 b20 5 1 a21b20  
a22 b16 2 1 a22b16  
a22 b23 3 1 a22b23  
a22 b28 3 1 a22b28  
a22 b21 4 1 a22b21  
a23 b17 2 1 a23b17  
a23 b29 1 1 a23b29  
a23 b22 1 1 a23b22  
a26 b20 5 1 a26b20  
a26 b27 2 1 a26b27  
a27 b21 4 1 a27b21  
a27 b28 3 1 a27b28  
a27 b26 2 1 a27b26  
a28 b22 1 1 a28b22  
a28 b29 1 1 a28b29  
a28 b27 2 1 a28b27  
a29 b23 3 1 a29b23  
a29 b28 3 1 a29b28

b8 d1 . 1 b8d1

b9 d1 . 1 b9d1

b10 d1 . 1 b10d1

b11 d1 . 1 b11d1

b14 d1 . 1 b14d1

b15 d1 . 1 b15d1

b16 d1 . 1 b16d1

b17 d1 . 1 b17d1

b20 d1 . 1 b20d1

b21 d1 . 1 b21d1

b22 d1 . 1 b22d1

b23 d1 . 1 b23d1

b26 d1 . 1 b26d1

b27 d1 . 1 b27d1

b28 d1 . 1 b28d1

b29 d1 . 1 b29d1

;

data cond;

input

ssa8	ssa9	ssa10	ssa11	ssa14	ssa15	ssa16	ssa17	ssa20	ssa21
ssa22	ssa23	ssa26	ssa27	ssa28	ssa29	a8b9	a8b14	a9b10	a9b15
a9b8	a10b11	a10b16	a10b9	a11b17	a11b10	a14b8	a14b15	a14b20	a15b9
a15b16	a15b21	a15b14	a16b10	a16b17	a16b22	a16b15	a17b11	a17b23	a17b16
a20b14	a20b21	a20b26	a21b15	a21b22	a21b27	a21b20	a22b16	a22b23	a22b28
a22b21	a23b17	a23b29	a23b22	a26b20	a26b27	a27b21	a27b28	a27b26	a28b22
a28b29	a28b27	a29b23	a29b28	b8d1	b9d1	b10d1	b11d1	b14d1	b15d1
b16d1	b17d1	b20d1	b21d1	b22d1	b23d1	b26d1	b27d1	b28d1	b29d1

\_type\_\$ \_rhs\_;

cards;

. . . . . 1 1 . . . . .  
. . . . . 1 . . . . .  
LE 1

. . . . . 1 1 1 . . . . .  
. . . . . 1 . . . . .  
LE 1

. . . . . 1 1 1 . . . . .  
. . . . . 1 . . . . .  
LE 1

. . . . . 1 1 . . . . .  
. . . . . 1 . . . . .  
LE 1

. . . . . 1 1 1 . . . . .  
. . . . . 1 . . . . .  
LE 1

. . . . . 1 1 1 1 . . . . .  
. . . . . 1 . . . . .  
LE 1

. . . . . 1 1 1 1 . . . . .  
. . . . . 1 . . . . .  
LE 1

. . . . . 1 1 1 . . . . .  
. . . . . 1 . . . . .  
LE 1

. . . . . 1 1 1 . . . . .  
. . . . . 1 . . . . .  
LE 1

. . . . . 1 1 1 1 . . . . .  
. . . . . 1 . . . . .  
LE 1

. . . . . 1 1 1 1 . . . . .  
. . . . . 1 . . . . .  
LE 1

. . . . . 1 1 1 . . . . .  
. . . . . 1 . . . . .  
LE 1



```

. . . . . 1 1 . . . . . 1 . .
LE 1

```

```

. . . . . 1 1 1 . . . . . 1 . .
LE 1

```

```

. . . . . 1 1 1 . . . . . 1 . .
LE 1

```

```

. . . . . 1 1 . . . . . 1
LE 1

```

```

;
proc netflow
  nodedata=noded
  arcdata=arcd
  conddata=cond
  conout=solution;
run;
print arcs/nonzero; run;

```

## C.5 Partial .log and .lis Output Files for Problem AREAWALL1

NOTE: Copyright(c) 1989 by SAS Institute Inc., Cary, NC USA.

NOTE: SAS (r) Proprietary Software Release 6.06.01

Licensed to AIR FORCE INSTITUTE OF TECHNOLOGY, Site 0003357011.

NOTE: Running on VAX Model 6000-420 Serial Number OB000005.

Welcome to the new SAS System, Release 6.06.

```
1 data noded;
2     input _node_ $ _sd_;
3     cards;
```

NOTE: The data set WORK.NODED has 2 observations and 2 variables.

```
6 ;
7 data arcd;
8     input _from_ $ _to_ $ _cost_ _capac_ _name_ $;
9     cards;
```

NOTE: The data set WORK.ARCDD has 80 observations and 5 variables.

```
90 ;
91 data cond;
92 input
93 ssa8      ssa9      ssa10     ssa11     ssa14     ssa15     ssa16     ssa17     ssa20     ssa21
94 ssa22     ssa23     ssa26     ssa27     ssa28     ssa29     a8b9      a8b14     a9b10     a9b15
95 a9b8      a10b11    a10b16    a10b9     a11b17    a11b10    a14b8     a14b15    a14b20    a15b9
96 a15b16    a15b21    a15b14    a16b10    a16b17    a16b22    a16b15    a17b11    a17b23    a17b16
97 a20b14    a20b21    a20b26    a21b15    a21b22    a21b27    a21b20    a22b16    a22b23    a22b28
98 a22b21    a23b17    a23b29    a23b22    a26b20    a26b27    a27b21    a27b28    a27b26    a28b22
99 a28b29    a28b27    a29b23    a29b28    b8d1      b9d1      b10d1     b11d1     b14d1     b15d1
100 b16d1     b17d1     b20d1     b21d1     b22d1     b23d1     b26d1     b27d1     b28d1     b29d1
101 _type_ $ _rhs_ ;
102 cards;
```

NOTE: SAS went to a new line when INPUT statement reached past the end of a line.

NOTE: The data set WORK.CONDD has 16 observations and 82 variables.

```
167 ;
168 proc netflow
169     nodedata=noded
170     arcdata=arcd
171     conddata=cond
172     conout=solution;
173 run;
```

NOTE: Number of nodes= 34 .

NOTE: Number of supply nodes= 1 .

NOTE: Number of demand nodes= 1 .  
 NOTE: Total supply= 1 , total demand= 1 .  
 NOTE: Number of arcs= 80 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 43 .  
 NOTE: Of these, 39 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 3 .  
 NOTE: Number of <= side constraints= 16 .  
 NOTE: Number of == side constraints= 0 .  
 NOTE: Number of >= side constraints= 0 .  
 NOTE: Number of arc and nonarc variable side constraint coefficients= 64 .  
 NOTE: Number of iterations, optimizing with constraints= 1 .  
 NOTE: Of these, 0 were degenerate.  
 NOTE: Optimum reached.  
 NOTE: Minimal total cost= 3 .  
 NOTE: The data set WORK.SOLUTION has 80 observations and 14 variables.  
 174            print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
1	ss	a22	1	1	0	SSA22	1
2	a22	b16	2	1	0	A22B16	.
3	b16	d1	0	1	0	B16D1	.

_N_	_DEMAND_	_FLOW_	_FCOST_	_RCOST_	_STATUS_
1	.	1	.	KEY_ARC	BASIC
2	.	1	2	KEY_ARC	BASIC
3	1	1	0	KEY_ARC	BASIC

## C.6 Partial .log and .lis Output Files for Problem AREAWALL2, Run 1

```
1 data noded;
2     input _node_ $ _sd_;
3     cards;
```

NOTE: The data set WORK.NODED has 2 observations and 2 variables.

```
6 ;
7 data arcd;
8     input _from_ $ _to_ $ _cost_ _capac_ _name_ $;
9     cards;
```

NOTE: The data set WORK.ARCN has 272 observations and 5 variables.

```
282 ;
283 data cond;
284 input
285 ssa8      ssa9      ssa10     ssa11     ssa14     ssa15     ssa16     ssa17     ssa20     ssa21
286 ssa22     ssa23     ssa26     ssa27     ssa28     ssa29     a8b9      a8b14     a9b10     a9b15
287 a9b8      a10b11    a10b16    a10b9     a11b17    a11b10    a14b8     a14b15    a14b20    a15b9
288 a15b16    a15b21    a15b14    a16b10    a16b17    a16b22    a16b15    a17b11    a17b23    a17b16
289 a20b14    a20b21    a20b26    a21b15    a21b22    a21b27    a21b20    a22b16    a22b23    a22b28
290 a22b21    a23b17    a23b29    a23b22    a26b20    a26b27    a27b21    a27b28    a27b26    a28b22
291 a28b29    a28b27    a29b23    a29b28    b8c9      b8c14     b9c10     b9c15     b9c8      b10c11
292 b10c16    b10c9     b11c17    b11c10    b14c8     b14c15    b14c20    b15c9     b15c16    b15c21
293 b15c14    b16c10    b16c17    b16c22    b16c15    b17c11    b17c23    b17c16    b20c14    b20c21
294 b20c26    b21c15    b21c22    b21c27    b21c20    b22c16    b22c23    b22c28    b22c21    b23c17
295 b23c29    b23c22    b26c20    b26c27    b27c21    b27c28    b27c26    b28c22    b28c29    b28c27
296 b29c23    b29c28    c8d9      c8d14     c9d10     c9d15     c9d8      c10d11    c10d16    c10d9
297 c11d17    c11d10    c14d8     c14d15    c14d20    c15d9     c15d16    c15d21    c15d14    c16d10
298 c16d17    c16d22    c16d15    c17d11    c17d23    c17d16    c20d14    c20d21    c20d26    c21d15
299 c21d22    c21d27    c21d20    c22d16    c22d23    c22d28    c22d21    c23d17    c23d29    c23d22
300 c26d20    c26d27    c27d21    c27d28    c27d26    c28d22    c28d29    c28d27    c29d23    c29d28
301 d8e9      d8e14     d9e10     d9e15     d9e8      d10e11    d10e16    d10e9     d11e17    d11e10
302 d14e8     d14e15    d14e20    d15e9     d15e16    d15e21    d15e14    d16e10    d16e17    d16e22
303 d16e15    d17e11    d17e23    d17e16    d20e14    d20e21    d20e26    d21e15    d21e22    d21e27
304 d21e20    d22e16    d22e23    d22e28    d22e21    d23e17    d23e29    d23e22    d26e20    d26e27
305 d27e21    d27e28    d27e26    d28e22    d28e29    d28e27    d29e23    d29e28    e8f9      e8f14
306 e9f10     e9f15     e9f8      e10f11    e10f16    e10f9     e11f17    e11f10    e14f8     e14f15
307 e14f20    e15f9     e15f16    e15f21    e15f14    e16f10    e16f17    e16f22    e16f15    e17f11
308 e17f23    e17f16    e20f14    e20f21    e20f26    e21f15    e21f22    e21f27    e21f20    e22f16
309 e22f23    e22f28    e22f21    e23f17    e23f29    e23f22    e26f20    e26f27    e27f21    e27f28
310 e27f26    e28f22    e28f29    e28f27    e29f23    e29f28    f8d1      f9d1      f10d1     f11d1
311 f14d1     f15d1     f16d1     f17d1     f20d1     f21d1     f22d1     f23d1     f26d1     f27d1
312 f28d1     f29d1     _type_ $ _rhs_ ;
313 cards;
```

NOTE: SAS went to a new line when INPUT statement reached past the end of a line.

NOTE: The data set WORK.COND has 32 observations and 274 variables.

```

586 ;
587 proc netflow
588     nodedata=noded
589     arcdata=arcd
590     condata=cond
591     conout=solution;
592 run;

```

```

NOTE: Number of nodes= 98 .
NOTE: Number of supply nodes= 1 .
NOTE: Number of demand nodes= 1 .
NOTE: Total supply= 1 , total demand= 1 .
NOTE: Number of arcs= 272 .
NOTE: Number of iterations performed (neglecting any constraints)= 59 .
NOTE: Of these, 58 were degenerate.
NOTE: Optimum (neglecting any constraints) found.
NOTE: Minimal total cost= 6 .
NOTE: Number of <= side constraints= 16 .
NOTE: Number of == side constraints= 0 .
NOTE: Number of >= side constraints= 16 .
NOTE: Number of arc and nonarc variable side constraint coefficients= 1320 .
NOTE: Number of iterations, optimizing with constraints= 808 .
NOTE: Of these, 791 were degenerate.
NOTE: Optimum reached.
NOTE: Minimal total cost= 6 .
NOTE: The data set WORK.SOLUTION has 272 observations and 14 variables.
593     print arcs/nonzero; run;

```

NOTE: The PROCEDURE NETFLOW printed pages 1-2.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
1	ss	a15	1	1	0	SSA15	1
2	ss	a16	1	1	0	SSA16	1
3	ss	a17	1	1	0	SSA17	1
4	ss	a23	1	1	0	SSA23	1
5	ss	a9	1	1	0	SSA9	1
6	a9	b15	1	1	0	A9B15	.
7	a17	b16	1	1	0	A17B16	.
8	a16	b22	1	1	0	A16B22	.
9	a23	b22	1	1	0	A23B22	.
10	a15	b9	1	1	0	A15B9	.
11	b9	c15	1	1	0	B9C15	.
12	b22	c16	1	1	0	B22C16	.
13	b16	c22	1	1	0	B16C22	.
14	b15	c9	1	1	0	B15C9	.
15	f8	d1	0	1	0	F8D1	.
16	f14	d1	0	1	0	F14D1	.
17	f16	d1	0	1	0	F16D1	.

18	f22	d1	0	1	0	F22D1
19	c15	d14	1	1	0	C15D14
20	c9	d15	1	1	0	C9D15
21	c16	d15	1	1	0	C16D15
22	c16	d22	1	1	0	C16D22
23	c22	d23	1	1	0	C22D23
24	d23	e17	1	1	0	D23E17
25	d22	e23	1	1	0	D22E23
26	d14	e8	1	1	0	D14E8
27	d15	e9	1	1	0	D15E9
28	e8	f14	1	1	0	E8F14
29	e17	f16	1	1	0	E17F16
30	e23	f22	1	1	0	E23F22
31	e9	f8	1	1	0	E9F8

<u>N</u>	<u>DEMAND</u>	<u>FLOW</u>	<u>FCOST</u>	<u>RCOST</u>	<u>STATUS</u>
1	.	0.25	0.25	.	KEY_ARC BASIC
2	.	0.0833333333	0.0833333333	.	KEY_ARC BASIC
3	.	0.25	0.25	.	NONKEY ARC BASIC
4	.	0.3333333333	0.3333333333	.	KEY_ARC BASIC
5	.	0.0833333333	0.0833333333	.	KEY_ARC BASIC
6	.	0.0833333333	0.0833333333	.	KEY_ARC BASIC
7	.	0.25	0.25	.	KEY_ARC BASIC
8	.	0.0833333333	0.0833333333	.	NONKEY ARC BASIC
9	.	0.3333333333	0.3333333333	.	KEY_ARC BASIC
10	.	0.25	0.25	.	KEY_ARC BASIC
11	.	0.25	0.25	.	NONKEY ARC BASIC
12	.	0.4166666667	0.4166666667	.	KEY_ARC BASIC
13	.	0.25	0.25	.	KEY_ARC BASIC
14	.	0.0833333333	0.0833333333	.	NONKEY ARC BASIC
15	1	0.4166666667	0	.	KEY_ARC BASIC
16	1	0.25	0	.	KEY_ARC BASIC
17	1	0.25	0	.	KEY_ARC BASIC
18	1	0.0833333333	0	.	KEY_ARC BASIC
19	.	0.25	0.25	.	KEY_ARC BASIC
20	.	0.0833333333	0.0833333333	.	KEY_ARC BASIC
21	.	0.3333333333	0.3333333333	.	KEY_ARC BASIC
22	.	0.0833333333	0.0833333333	.	NONKEY ARC BASIC
23	.	0.25	0.25	.	KEY_ARC BASIC
24	.	0.25	0.25	.	KEY_ARC BASIC
25	.	0.0833333333	0.0833333333	.	KEY_ARC BASIC
26	.	0.25	0.25	.	KEY_ARC BASIC
27	.	0.4166666667	0.4166666667	.	KEY_ARC BASIC
28	.	0.25	0.25	.	KEY_ARC BASIC
29	.	0.25	0.25	.	KEY_ARC BASIC
30	.	0.0833333333	0.0833333333	.	KEY_ARC BASIC
31	.	0.4166666667	0.4166666667	.	KEY_ARC BASIC

*C:7 Partial .log and .lis Output Files for Problem AREAWALL2, Run 2*

NOTE: Number of nodes= 98 .  
 NOTE: Number of supply nodes= 1 .  
 NOTE: Number of demand nodes= 1 .  
 NOTE: Total supply= 1 , total demand= 1 .  
 NOTE: Number of arcs= 272 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 59 .  
 NOTE: Of these, 58 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 6 .  
 NOTE: Number of <= side constraints= 16 .  
 NOTE: Number of == side constraints= 1 .  
 NOTE: Number of >= side constraints= 16 .  
 NOTE: Number of arc and nonarc variable side constraint coefficients= 1321 .  
 NOTE: Number of iterations, optimizing with constraints= 938 .  
 NOTE: Of these, 923 were degenerate.  
 NOTE: Optimum reached.  
 NOTE: Minimal total cost= 6.5 .  
 NOTE: The data set WORK.SOLUTION has 272 observations and 14 variables.  
 601 print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
1	ss	a23	1	1	0	SSA23	1
2	a23	b22	1	1	0	A23B22	.
3	b22	c16	1	1	0	B22C16	.
4	b22	c21	2	1	0	B22C21	.
5	f8	d1	0	1	0	F8D1	.
6	f15	d1	0	1	0	F15D1	.
7	f17	d1	0	1	0	F17D1	.
8	c21	d15	1	1	0	C21D15	.
9	c16	d17	1	1	0	C16D17	.
10	d15	e14	1	1	0	D15E14	.
11	d17	e16	1	1	0	D17E16	.
12	e14	f15	1	1	0	E14F15	.
13	e16	f17	1	1	0	E16F17	.
14	e14	f8	1	1	0	E14F8	.

_N_	_DEMAND_	_FLOW_	_FCOST_	_RCOST_	_STATUS_
1	.	1	1	.	KEY_ARC BASIC
2	.	1	1	.	KEY_ARC BASIC
3	.	0.5	0.5	.	NONKEY ARC BASIC
4	.	0.5	1	.	NONKEY ARC BASIC
5	1	0.25	0	.	KEY_ARC BASIC
6	1	0.25	0	.	NONKEY ARC BASIC

7	1	0.5	0	.	KEY_ARC	BASIC
8	.	0.5	0.5	.	KEY_ARC	BASIC
9	.	0.5	0.5	.	KEY_ARC	BASIC
10	.	0.5	0.5	.	KEY_ARC	BASIC
11	.	0.5	0.5	.	KEY_ARC	BASIC
12	.	0.25	0.25	.	KEY_ARC	BASIC
13	.	0.5	0.5	.	KEY_ARC	BASIC
14	.	0.25	0.25	.	NONKEY ARC	BASIC



# C:8 Partial .log and .lis Output Files for Problem AREAWALL2, Run 3

NOTE: Number of nodes= 98 .  
 NOTE: Number of supply nodes= 1 .  
 NOTE: Number of demand nodes= 1 .  
 NOTE: Total supply= 1 , total demand= 1 .  
 NOTE: Number of arcs= 272 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 59 .  
 NOTE: Of these, 58 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 6 .  
 NOTE: Number of <= side constraints= 16 .  
 NOTE: Number of == side constraints= 1 .  
 NOTE: Number of >= side constraints= 16 .  
 NOTE: Number of arc and nonarc variable side constraint coefficients= 1322 .  
 NOTE: Number of iterations, optimizing with constraints= 865 .  
 NOTE: Of these, 851 were degenerate.  
 NOTE: Optimum reached.  
 NOTE: Minimal total cost= 9 .  
 NOTE: The data set WORK.SOLUTION has 272 observations and 14 variables.  
 601 print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
1	ss	a23	1	1	0	SSA23	1
2	a23	b22	1	1	0	A23B22	.
3	b22	c16	1	1	0	B22C16	.
4	f8	d1	0	1	0	F8D1	.
5	f15	d1	0	1	0	F15D1	.
6	f17	d1	0	1	0	F17D1	.
7	c16	d15	1	1	0	C16D15	.
8	c16	d17	1	1	0	C16D17	.
9	d17	e11	7	1	0	D17E11	.
10	d15	e9	1	1	0	D15E9	.
11	e9	f15	1	1	0	E9F15	.
12	e11	f17	1	1	0	E11F17	.
13	e9	f8	1	1	0	E9F8	.

_N_	_DEMAND_	_FLOW_	_FCOST_	_RCOST_	_STATUS_
1	.	1	1	.	KEY_ARC BASIC
2	.	1	1	.	NONKEY ARC BASIC
3	.	1	1	.	KEY_ARC BASIC
4	1	0.25	0	.	KEY_ARC BASIC
5	1	0.25	0	.	NONKEY ARC BASIC
6	1	0.5	0	.	KEY_ARC BASIC
7	.	0.5	0.5	.	KEY_ARC BASIC

8	.	0.5	0.5	.	NONKEY ARC BASIC
9	.	0.5	3.5	.	KEY_ARC BASIC
10	.	0.5	0.5	.	KEY_ARC BASIC
11	.	0.25	0.25	.	KEY_ARC BASIC
12	.	0.5	0.5	.	KEY_ARC BASIC
13	.	0.25	0.25	.	KEY_ARC BASIC

### C.9 Partial .log and .lis Output Files for Problem AREAWALL2, Run 4

NOTE: Number of nodes= 98 .  
 NOTE: Number of supply nodes= 1 .  
 NOTE: Number of demand nodes= 1 .  
 NOTE: Total supply= 1 , total demand= 1 .  
 NOTE: Number of arcs= 272 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 59 .  
 NOTE: Of these, 58 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 6 .  
 NOTE: Number of <= side constraints= 16 .  
 NOTE: Number of == side constraints= 1 .  
 NOTE: Number of >= side constraints= 16 .  
 NOTE: Number of arc and nonarc variable side constraint coefficients= 1323 .  
 NOTE: Number of iterations, optimizing with constraints= 753 .  
 NOTE: Of these, 735 were degenerate.  
 WARNING: The following constraints are not satisfied. The value after each constraint name is the amount by which the constraint is violated; ie. the value of the corresponding artificial variable.       \_OBS33\_ 0.5  
 NOTE: The problem is infeasible. At least one side constraint cannot be satisfied.  
 WARNING: The solution value reported next is associated with a solution that is not both feasible and optimal.  
 NOTE: Minimal total cost= 7 .  
 NOTE: The data set WORK.SOLUTION has 272 observations and 14 variables.  
 601       print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
1	ss	a21	2	1	0	SSA21	1
2	ss	a23	1	1	0	SSA23	1
3	a21	b22	1	1	0	A21B22	.
4	a23	b22	1	1	0	A23B22	.
5	b22	c16	1	1	0	B22C16	.
6	f8	d1	0	1	0	F8D1	.
7	f20	d1	0	1	0	F20D1	.
8	c16	d15	1	1	0	C16D15	.
9	d15	e14	1	1	0	D15E14	.
10	d15	e9	1	1	0	D15E9	.
11	e14	f20	2	1	0	E14F20	.
12	e9	f8	1	1	0	E9F8	.

_N_	_DEMAND_	_FLOW_	_FCOST_	_RCOST_	_STATUS_
1	.	0.5	1	.	KEY_ARC BASIC
2	.	0.5	0.5	.	KEY_ARC BASIC
3	.	0.5	0.5	.	NONKEY ARC BASIC

4	.	0.5	0.5	.	KEY_ARC	BASIC
5	.	1	1	.	KEY_ARC	BASIC
6	1	0.5	0	.	KEY_ARC	BASIC
7	1	0.5	0	.	KEY_ARC	BASIC
8	.	1	1	.	KEY_ARC	BASIC
9	.	0.5	0.5	.	KEY_ARC	BASIC
10	.	0.5	0.5	.	NONKEY ARC	BASIC
11	.	0.5	1	.	KEY_ARC	BASIC
12	.	0.5	0.5	.	KEY_ARC	BASIC

# G.10 Partial .log and .lis Output Files for Problem AREAWALL2, Run 5

NOTE: Number of nodes= 98 .  
 NOTE: Number of supply nodes= 1 .  
 NOTE: Number of demand nodes= 1 .  
 NOTE: Total supply= 1 , total demand= 1 .  
 NOTE: Number of arcs= 272 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 59 .  
 NOTE: Of these, 58 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 6 .  
 NOTE: Number of <= side constraints= 16 .  
 NOTE: Number of == side constraints= 1 .  
 NOTE: Number of >= side constraints= 16 .  
 NOTE: Number of arc and nonarc variable side constraint coefficients= 1323 .  
 NOTE: Number of iterations, optimizing with constraints= 730 .  
 NOTE: Of these, 685 were degenerate.  
 NOTE: Optimum reached.  
 NOTE: Minimal total cost= 18 .  
 NOTE: The data set WORK.SOLUTION has 272 observations and 14 variables.  
 601 print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
1	ss	a23	1	1	0	SSA23	1
2	a23	b22	1	1	0	A23B22	.
3	b22	c16	1	1	0	B22C16	.
4	f10	d1	0	1	0	F10D1	.
5	c16	d17	1	1	0	C16D17	.
6	d17	e11	7	1	0	D17E11	.
7	e11	f10	7	1	0	E11F10	.

_N_	_DEMAND_	_FLOW_	_FCOST_	_RCOST_	_STATUS_
1	.	1	1	.	KEY_ARC BASIC
2	.	1	1	.	KEY_ARC BASIC
3	.	1	1	.	NONKEY ARC BASIC
4	1	1	0	.	NONKEY ARC BASIC
5	.	1	1	0	UPPERBD NONBASIC
6	.	1	7	.	KEY_ARC BASIC
7	.	1	7	.	KEY_ARC BASIC

C:11 Partial .log and .lis Output Files for Problem AREAWALL2, Run 8

NOTE: Number of nodes= 98 .  
 NOTE: Number of supply nodes= 1 .  
 NOTE: Number of demand nodes= 1 .  
 NOTE: Total supply= 1 , total demand= 1 .  
 NOTE: Number of arcs= 272 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 59 .  
 NOTE: Of these, 58 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 6 .  
 NOTE: Number of <= side constraints= 16 .  
 NOTE: Number of == side constraints= 1 .  
 NOTE: Number of >= side constraints= 16 .  
 NOTE: Number of arc and nonarc variable side constraint coefficients= 1322 .  
 NOTE: Number of iterations, optimizing with constraints= 789 .  
 NOTE: Of these, 766 were degenerate.  
 NOTE: Optimum reached.  
 NOTE: Minimal total cost= 7 .  
 NOTE: The data set WORK.SOLUTION has 272 observations and 14 variables.  
 601 print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
1	ss	a15	1	1	0	SSA15	1
2	a15	b16	1	1	0	A15B16	.
3	b16	c17	1	1	0	B16C17	.
4	b16	c22	1	1	0	B16C22	.
5	f21	d1	0	1	0	F21D1	.
6	f23	d1	0	1	0	F23D1	.
7	c22	d21	2	1	0	C22D21	.
8	c17	d23	1	1	0	C17D23	.
9	d23	e17	1	1	0	D23E17	.
10	d21	e22	1	1	0	D21E22	.
11	e22	f21	2	1	0	E22F21	.
12	e17	f23	1	1	0	E17F23	.

_N_	_DEMAND_	_FLOW_	_FCOST_	_RCOST_	_STATUS_
1	.	1	1	-2	UPPERBD NONBASIC
2	.	1	1	.	KEY_ARC BASIC
3	.	0.5	0.5	.	KEY_ARC BASIC
4	.	0.5	0.5	.	KEY_ARC BASIC
5	1	0.5	0	.	KEY_ARC BASIC
6	1	0.5	0	.	KEY_ARC BASIC
7	.	0.5	1	.	KEY_ARC BASIC
8	.	0.5	0.5	.	NONKEY ARC BASIC

9	.	0.5	0.5	.	KEY_ARC	BASIC
10	.	0.5	0.5	.	KEY_ARC	BASIC
11	.	0.5	1	.	NONKEY ARC	BASIC
12	.	0.5	0.5	.	KEY_ARC	BASIC

## C.12 Partial .log and .lis Output Files for Problem AREAWALL3, Run 1

```

1 data noded;
2     input _node_ $ _sd_;
3     cards;

```

NOTE: The data set WORK.NODED has 3 observations and 2 variables.

```

7 ;
8 data arcd;
9     input _from_ $ _to_ $ _cost_ _capac_ _name_ $;
10    cards;

```

NOTE: The data set WORK.ARCN has 288 observations and 5 variables.

```

299 ;
300 data cond;
301 input
302 ssa8      ssa9      ssa10     ssa11     ssa14     ssa15     ssa16     ssa17     ssa20     ssa21
303 ssa22     ssa23     ssa26     ssa27     ssa28     ssa29     a8b9      a8b14     a9b10     a9b15
304 a9b8      a10b11    a10b16    a10b9     a11b17    a11b10    a14b8      a14b15    a14b20    a15b9
305 a15b16    a15b21    a15b14    a16b10    a16b17    a16b22    a16b15    a17b11    a17b23    a17b16
306 a20b14    a20b21    a20b26    a21b15    a21b22    a21b27    a21b20    a22b16    a22b23    a22b28
307 a22b21    a23b17    a23b29    a23b22    a26b20    a26b27    a27b21    a27b28    a27b26    a28b22
308 a28b29    a28b27    a29b23    a29b28    b8c9      b8c14     b9c10     b9c15     b9c8      b10c11
309 b10c16    b10c9     b11c17    b11c10    b14c8     b14c15    b14c20    b15c9     b15c16    b15c21
310 b15c14    b16c10    b16c17    b16c22    b16c15    b17c11    b17c23    b17c16    b20c14    b20c21
311 b20c26    b21c15    b21c22    b21c27    b21c20    b22c16    b22c23    b22c28    b22c21    b23c17
312 b23c29    b23c22    b26c20    b26c27    b27c21    b27c28    b27c26    b28c22    b28c29    b28c27
313 b29c23    b29c28    c8d9      c8d14     c9d10     c9d15     c9d8      c10d11    c10d16    c10d9
314 c11d17    c11d10    c14d8     c14d15    c14d20    c15d9     c15d16    c15d21    c15d14    c16d10
315 c16d17    c16d22    c16d15    c17d11    c17d23    c17d16    c20d14    c20d21    c20d26    c21d15
316 c21d22    c21d27    c21d20    c22d16    c22d23    c22d28    c22d21    c23d17    c23d29    c23d22
317 c26d20    c26d27    c27d21    c27d28    c27d26    c28d22    c28d29    c28d27    c29d23    c29d28
318 d8e9      d8e14     d9e10     d9e15     d9e8      d10e11    d10e16    d10e9     d11e17    d11e10
319 d14e8     d14e15    d14e20    d15e9     d15e16    d15e21    d15e14    d16e10    d16e17    d16e22
320 d16e15    d17e11    d17e23    d17e16    d20e14    d20e21    d20e26    d21e15    d21e22    d21e27
321 d21e20    d22e16    d22e23    d22e28    d22e21    d23e17    d23e29    d23e22    d26e20    d26e27
322 d27e21    d27e28    d27e26    d28e22    d28e29    d28e27    d29e23    d29e28    e8f9      e8f14
323 e9f10     e9f15     e9f8      e10f11    e10f16    e10f9     e11f17    e11f10    e14f8     e14f15
324 e14f20    e15f9     e15f16    e15f21    e15f14    e16f10    e16f17    e16f22    e16f15    e17f11
325 e17f23    e17f16    e20f14    e20f21    e20f26    e21f15    e21f22    e21f27    e21f20    e22f16
326 e22f23    e22f28    e22f21    e23f17    e23f29    e23f22    e26f20    e26f27    e27f21    e27f28
327 e27f26    e28f22    e28f29    e28f27    e29f23    e29f28    f8d1      f9d1      f10d1     f11d1
328 f14d1     f15d1     f16d1     f17d1     f20d1     f21d1     f22d1     f23d1     f26d1     f27d1
329 f28d1     f29d1     d8d2      d9d2      d10d2     d11d2     d14d2     d15d2     d16d2     d17d2
330 d20d2     d21d2     d22d2     d23d2     d26d2     d27d2     d28d2     d29d2     _type_ $ _rhs_ ;
331 cards;

```

NOTE: SAS went to a new line when INPUT statement reached past the end of a line.

NOTE: The data set WORK.COND has 32 observations and 290 variables.



```

620 ;
621 proc netflow
622     nodedata=noded
623     arcdata=arcd
624     condata=cond
625     conout=solution;
626 run;

```

```

NOTE: Number of nodes= 99 .
NOTE: Number of supply nodes= 1 .
NOTE: Number of demand nodes= 2 .
NOTE: Total supply= 2 , total demand= 2 .
NOTE: Number of arcs= 288 .
NOTE: Number of iterations performed (neglecting any constraints)= 98 .
NOTE: Of these, 96 were degenerate.
NOTE: Optimum (neglecting any constraints) found.
NOTE: Minimal total cost= 10 .
NOTE: Number of <= side constraints= 16 .
NOTE: Number of == side constraints= 0 .
NOTE: Number of >= side constraints= 16 .
NOTE: Number of arc and nonarc variable side constraint coefficients= 1400 .
NOTE: Number of iterations, optimizing with constraints= 484 .
NOTE: Of these, 470 were degenerate.
NOTE: Optimum reached.
NOTE: Minimal total cost= 12 .
NOTE: The data set WORK.SOLUTION has 288 observations and 14 variables.
627     print arcs/nonzero; run;

```

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
1	ss	a14	1	1	0	SSA14	2
2	ss	a17	1	1	0	SSA17	2
3	ss	a21	2	1	0	SSA21	2
4	ss	a27	2	1	0	SSA27	2
5	a21	b15	1	1	0	A21B15	.
6	a17	b16	1	1	0	A17B16	.
7	a14	b20	2	1	0	A14B20	.
8	a27	b21	2	1	0	A27B21	.
9	b20	c14	1	1	0	B20C14	.
10	b21	c15	1	1	0	B21C15	.
11	b15	c21	2	1	0	B15C21	.
12	b16	c22	1	1	0	B16C22	.
13	f8	d1	0	1	0	F8D1	.
14	f9	d1	0	1	0	F9D1	.
15	c14	d15	1	1	0	C14D15	.
16	d23	d2	0	1	0	D23D2	.

17	c21	d20	2	1	0	C21D20	.
18	c22	d23	1	1	0	C22D23	.
19	c15	d9	1	1	0	C15D9	.
20	d15	e14	1	1	0	D15E14	.
21	d20	e14	1	1	0	D20E14	.
22	d9	e8	1	1	0	D9E8	.
23	e14	f8	1	1	0	E14F8	.
24	e8	f9	1	1	0	E8F9	.

<u>_N_</u>	<u>_DEMAND_</u>	<u>_FLOW_</u>	<u>_FCOST_</u>	<u>_RCOST_</u>	<u>_STATUS_</u>
1	.	0.25	0.25	.	KEY_ARC BASIC
2	.	1	1	.	NONKEY ARC BASIC
3	.	0.25	0.5	.	NONKEY ARC BASIC
4	.	0.5	1	.	NONKEY ARC BASIC
5	.	0.25	0.25	.	KEY_ARC BASIC
6	.	1	1	.	KEY_ARC BASIC
7	.	0.25	0.5	.	NONKEY ARC BASIC
8	.	0.5	1	.	KEY_ARC BASIC
9	.	0.25	0.25	.	KEY_ARC BASIC
10	.	0.5	0.5	.	KEY_ARC BASIC
11	.	0.25	0.5	.	NONKEY ARC BASIC
12	.	1	1	.	KEY_ARC BASIC
13	1	0.5	0	.	KEY_ARC BASIC
14	1	0.5	0	.	KEY_ARC BASIC
15	.	0.25	0.25	.	KEY_ARC BASIC
16	1	1	0	.	KEY_ARC BASIC
17	.	0.25	0.5	.	KEY_ARC BASIC
18	.	1	1	.	KEY_ARC BASIC
19	.	0.5	0.5	.	KEY_ARC BASIC
20	.	0.25	0.25	.	KEY_ARC BASIC
21	.	0.25	0.25	.	KEY_ARC BASIC
22	.	0.5	0.5	.	NONKEY ARC BASIC
23	.	0.5	0.5	.	KEY_ARC BASIC
24	.	0.5	0.5	.	KEY_ARC BASIC

### C.13: Partial .log and .lis Output Files for Problem AREAWALL3, Run 2

NOTE: Number of nodes= 99 .  
 NOTE: Number of supply nodes= 1 .  
 NOTE: Number of demand nodes= 2 .  
 NOTE: Total supply= 2 , total demand= 2 .  
 NOTE: Number of arcs= 288 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 98 .  
 NOTE: Of these, 96 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 10 .  
 NOTE: Number of <= side constraints= 16 .  
 NOTE: Number of == side constraints= 1 .  
 NOTE: Number of >= side constraints= 16 .  
 NOTE: Number of arc and nonarc variable side constraint coefficients= 1402 .  
 NOTE: Number of iterations, optimizing with constraints= 768 .  
 NOTE: Of these, 748 were degenerate.  
 NOTE: Optimum reached.  
 NOTE: Minimal total cost= 12 .  
 NOTE: The data set WORK.SOLUTION has 288 observations and 14 variables.  
 636        print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
1	ss	a17	1	1	0	SSA17	2
2	ss	a8	1	1	0	SSA8	2
3	a8	b14	1	1	0	A8B14	.
4	a17	b16	1	1	0	A17B16	.
5	a17	b23	1	1	0	A17B23	.
6	b14	c15	1	1	0	B14C15	.
7	b16	c15	1	1	0	B16C15	.
8	b14	c20	2	1	0	B14C20	.
9	b23	c22	1	1	0	B23C22	.
10	f16	d1	0	1	0	F16D1	.
11	f21	d1	0	1	0	F21D1	.
12	d9	d2	0	1	0	D9D2	.
13	c20	d21	2	1	0	C20D21	.
14	c22	d23	1	1	0	C22D23	.
15	c15	d9	1	1	0	C15D9	.
16	d23	e22	1	1	0	D23E22	.
17	d21	e27	2	1	0	D21E27	.
18	e22	f16	1	1	0	E22F16	.
19	e27	f21	2	1	0	E27F21	.

_N_	_DEMAND_	_FLOW_	_FCOST_	_RCOST_	_STATUS_
1	.	1	1	.	KEY_ARC BASIC

2	.	1	1	-99999987	UPPERBD NONBASIC
3	.	1	1	.	KEY_ARC BASIC
4	.	0.5	0.5	.	KEY_ARC BASIC
5	.	0.5	0.5	.	KEY_ARC BASIC
6	.	0.5	0.5	.	KEY_ARC BASIC
7	.	0.5	0.5	.	KEY_ARC BASIC
8	.	0.5	1	.	KEY_ARC BASIC
9	.	0.5	0.5	.	KEY_ARC BASIC
10	1	0.5	0	.	KEY_ARC BASIC
11	1	0.5	0	.	NONKEY ARC BASIC
12	1	1	0	.	KEY_ARC BASIC
13	.	0.5	1	.	KEY_ARC BASIC
14	.	0.5	0.5	.	KEY_ARC BASIC
15	.	1	1	.	KEY_ARC BASIC
16	.	0.5	0.5	.	KEY_ARC BASIC
17	.	0.5	1	.	KEY_ARC BASIC
18	.	0.5	0.5	.	NONKEY ARC BASIC
19	.	0.5	1	.	NONKEY ARC BASIC

# C.14 Partial .log and .lis Output Files for Problem AREAWALL3, Run 3

NOTE: Number of nodes= 99 .  
 NOTE: Number of supply nodes= 1 .  
 NOTE: Number of demand nodes= 2 .  
 NOTE: Total supply= 2 , total demand= 2 .  
 NOTE: Number of arcs= 288 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 98 .  
 NOTE: Of these, 96 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 10 .  
 NOTE: Number of <= side constraints= 16 .  
 NOTE: Number of == side constraints= 1 .  
 NOTE: Number of >= side constraints= 16 .  
 NOTE: Number of arc and nonarc variable side constraint coefficients= 1407 .  
 NOTE: Number of iterations, optimizing with constraints= 1180 .  
 NOTE: Of these, 1155 were degenerate.  
 NOTE: Optimum reached.  
 NOTE: Minimal total cost= 12 .  
 NOTE: The data set WORK.SOLUTION has 288 observations and 14 variables.  
 636        print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
1	ss	a17	1	1	0	SSA17	2
2	ss	a8	1	1	0	SSA8	2
3	a8	b14	1	1	0	A8B14	.
4	a17	b16	1	1	0	A17B16	.
5	b14	c20	2	1	0	B14C20	.
6	b16	c22	1	1	0	B16C22	.
7	f9	d1	0	1	0	F9D1	.
8	d23	d2	0	1	0	D23D2	.
9	c20	d21	2	1	0	C20D21	.
10	c22	d23	1	1	0	C22D23	.
11	d21	e15	1	1	0	D21E15	.
12	e15	f9	1	1	0	E15F9	.

_N_	_DEMAND_	_FLOW_	_FCOST_	_RCOST_	_STATUS_
1	.	1	1	.	KEY_ARC BASIC
2	.	1	1	.	KEY_ARC BASIC
3	.	1	1	.	KEY_ARC BASIC
4	.	1	1	.	KEY_ARC BASIC
5	.	1	2	.	KEY_ARC BASIC
6	.	1	1	.	KEY_ARC BASIC
7	1	1	0	.	KEY_ARC BASIC
8	1	1	0	0	UPPERBD NONBASIC

9	.	1	2	.	KEY_ARC	BASIC
10	.	1	1	.	KEY_ARC	BASIC
11	.	1	1	1.110223E-16	UPPERBD	NONBASIC
12	.	1	1	.	KEY_ARC	BASIC

# C.15 Partial .log and .lis Output Files for Problem CELLBORDER1, Run 1

NOTE: Number of nodes= 108 .  
 NOTE: Number of supply nodes= 1 .  
 NOTE: Number of demand nodes= 2 .  
 NOTE: Total supply= 8 , total demand= 8 .  
 NOTE: Number of arcs= 216 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 128 .  
 NOTE: Of these, 117 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 8 .  
 NOTE: The data set WORK.SOLUTION has 216 observations and 14 variables.  
 234 print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_SUPPLY_	_DEMAND_
1	b17	DE	0	1	0	.	6
2	b18	DE	0	1	0	.	6
3	b19	DE	0	1	0	.	6
4	b20	DE	0	1	0	.	6
5	b41	DE	0	1	0	.	6
6	b42	DE	0	1	0	.	6
7	c20	DIC	0	2	0	.	2
8	c21	DIC	0	2	0	.	2
9	m14	b17	0	1	0	.	.
10	m14	b18	0	1	0	.	.
11	m14	b19	0	1	0	.	.
12	m14	b20	0	1	0	.	.
13	m22	b41	0	1	0	.	.
14	m22	b42	0	1	0	.	.
15	m22	b43	0	1	0	.	.
16	m29	b61	0	1	0	.	.
17	b43	c20	0	1	0	.	.
18	b61	c21	0	1	0	.	.
19	S	m14	1	4	0	8	.
20	S	m22	1	4	0	8	.
21	S	m29	1	4	0	8	.

_N_	_FLOW_	_FCOST_	_RCOST_	_STATUS_	_NAME_
1	1	0	0	UPPERBD NONBASIC	b17DE
2	1	0	0	UPPERBD NONBASIC	b18DE
3	1	0	.	KEY_ARC BASIC	b19DE
4	1	0	0	UPPERBD NONBASIC	b20DE
5	1	0	.	KEY_ARC BASIC	b41DE
6	1	0	0	UPPERBD NONBASIC	b42DE
7	1	0	.	KEY_ARC BASIC	c20DIC

8	1	0
9	1	0
10	1	0
11	1	0
12	1	0
13	1	0
14	1	0
15	1	0
16	1	0
17	1	0
18	1	0
19	4	4
20	3	3
21	1	1

.	KEY_ARC	BASIC	c21DIC
0	UPPERBD	NONBASIC	m14b17
.	KEY_ARC	BASIC	m14b18
0	UPPERBD	NONBASIC	m14b19
.	KEY_ARC	BASIC	m14b20
.	UPPERBD	NONBASIC	m22b41
.	KEY_ARC	BASIC	m22b42
.	KEY_ARC	BASIC	m22b43
.	KEY_ARC	BASIC	m29b61
0	UPPERBD	NONBASIC	b43c20
.	KEY_ARC	BASIC	b61c21
.	KEY_ARC	BASIC	Sm14
.	KEY_ARC	BASIC	Sm22
.	KEY_ARC	BASIC	Sm29



### C.16 Input File for Problem CELLBORDER1, Run 2

```

data nodeu;
    input _node_$ _sd_;
cards;
S      8
DE     -6
DIR     0
DIC     -2
;
data arcd;
input _from_$ _to_$ _cost_ _capac_ _name_$;
cards;
S m8 . 4 Sm8
S m9 . 4 Sm9
S m10 . 4 Sm10
S m11 . 4 Sm11
S m14 . 4 Sm14
S m15 . 4 Sm15
S m16 . 4 Sm16
S m17 . 4 Sm17
S m20 . 4 Sm20
S m21 . 4 Sm21
S m22 . 4 Sm22
S m23 . 4 Sm23
S m26 . 4 Sm26
S m27 . 4 Sm27
S m28 . 4 Sm28
S m29 . 4 Sm29
m8 b1 . 1 m8b1
b1 DE 6 1 b1DE
m8 b2 . 1 m8b2
b2 DE 6 1 b2DE
b2 c1 6 1 b2c1
c1 DIR . 2 c1DIR
m8 b3 . 1 m8b3
b3 DE 6 1 b3DE
b3 c2 . 1 b3c2
c2 DIC . 2 c2DIC
m8 b4 . 1 m8b4
b4 DE 6 1 b4DE
m9 b5 . 1 m9b5
b5 DE 5 1 b5DE
m9 b6 . 1 m9b6
b6 DE 5 1 b6DE
b6 c3 5 1 b6c3
c3 DIR . 2 c3DIR
m9 b7 . 1 m9b7
b7 DE 5 1 b7DE
b7 c4 5 1 b7c4
c4 DIC . 2 c4DIC
m9 b8 . 1 m9b8

```

b8 DE 5 1 b8DE  
 b8 c1 5 1 b8c1  
 m10 b9 . 1 m10b9  
 b9 DE 3 1 b9DE  
 m10 b10 . 1 m10b10  
 b10 DE 3 1 b10DE  
 b10 c5 3 1 b10c5  
 c5 DIR . 2 c5DIR  
 m10 b11 . 1 m10b11  
 b11 DE 3 1 b11DE  
 b11 c6 3 1 b11c6  
 c6 DIC . 2 c6DIC  
 m10 b12 . 1 m10b12  
 b12 DE 3 1 b12DE  
 b12 c3 3 1 b12c3  
 m11 b13 . 1 m11b13  
 b13 DE 4 1 b13DE  
 m11 b14 . 1 m11b14  
 b14 DE 4 1 b14DE  
 m11 b15 . 1 m11b15  
 b15 DE 4 1 b15DE  
 b15 c7 4 1 b15c7  
 c7 DIC . 2 c7DIC  
 m11 b16 . 1 m11b16  
 b16 DE 4 1 b16DE  
 b16 c5 4 1 b16c5  
 m14 b17 . 1 m14b17  
 b17 DE 1 1 b17DE  
 b17 c2 1 1 b17c2  
 m11 b18 . 1 m14b18  
 b18 DE 1 1 b18DE  
 b18 c8 1 1 b18c8  
 c8 DIR . 2 c8DIR  
 m14 b19 . 1 m14b19  
 b19 DE 1 1 b19DE  
 b19 c9 1 1 b19c9  
 c9 DIC . 2 c9DIC  
 m14 b20 . 1 m14b20  
 b20 DE 1 1 b20DE  
 m15 b21 . 1 m15b21  
 b21 DE 3 1 b21DE  
 b21 c4 3 1 b21c4  
 m15 b22 . 1 m15b22  
 b22 DE 3 1 b22DE  
 b22 c10 3 1 b22c10  
 c10 DIR . 2 c10DIR  
 m15 b23 . 1 m15b23  
 b23 DE 3 1 b23DE  
 b23 c11 3 1 b23c11  
 c11 DIC . 2 c11DIC  
 m15 b24 . 1 m15b24

b24	DE	3	1	b24DE
b24	c8	3	1	b24c8
m16	b25	.	1	m16b25
b25	DE	2	1	b25DE
b25	c6	2	1	b25c6
m16	b26	.	1	m16b26
b26	DE	2	1	b26DE
b26	c12	2	1	b26c12
c12	DIR	.	2	c12DIR
m16	b27	.	1	m16b27
b27	DE	2	1	b27DE
b27	c13	2	1	b27c13
c13	DIC	.	2	c13DIC
m16	b28	.	1	m16b28
b28	DE	2	1	b28DE
b28	c10	2	1	b28c10
m17	b29	.	1	m17b29
b29	DE	2	1	b29DE
b29	c7	2	1	b29c7
m17	b30	.	1	m17b30
b30	DE	2	1	b30DE
m17	b31	.	1	m17b31
b31	DE	2	1	b31DE
b31	c14	2	1	b31c14
c14	DIC	.	2	c14DIC
m17	b32	.	1	m17b32
b32	DE	2	1	b32DE
b32	c12	2	1	b32c12
m20	b33	.	1	m20b33
b33	DE	5	1	b33DE
b33	c9	5	1	b33c9
m20	b34	.	1	m20b34
b34	DE	5	1	b34DE
b34	c15	5	1	b34c15
c15	DIR	.	2	c15DIR
m20	b35	.	1	m20b35
b35	DE	5	1	b35DE
b35	c16	5	1	b35c16
c16	DIC	.	2	c16DIC
m20	b36	.	1	m20b36
b36	DE	5	1	b36DE
m21	b37	.	1	m21b37
b37	DE	4	1	b37DE
b37	c11	4	1	b37c11
m21	b38	.	1	m21b38
b38	DE	4	1	b38DE
b38	c17	4	1	b38c17
c17	DIR	.	2	c17DIR
m21	b39	.	1	m21b39
b39	DE	4	1	b39DE
b39	c18	4	1	b39c18

c18	DIC	.	2	c18DIC
m21	b40	.	1	m21b40
b40	DE	4	1	b40DE
b40	c15	4	1	b40c15
m22	b41	.	1	m22b41
b41	DE	1	1	b41DE
b41	c13	1	1	b41c13
m22	b42	.	1	m22b42
b42	DE	1	1	b42DE
b42	c19	1	1	b42c19
c19	DIR	.	2	c19DIR
m22	b43	.	1	m22b43
b43	DE	1	1	b43DE
b43	c20	1	1	b43c20
c20	DIC	.	2	c20DIC
m22	b44	.	1	m22b44
b44	DE	1	1	b44DE
b44	c17	1	1	b44c17
m23	b45	.	1	m23b45
b45	DE	3	1	b45DE
b45	c14	3	1	b45c14
m23	b46	.	1	m23b46
b46	DE	3	1	b46DE
m23	b47	.	1	m23b47
b47	DE	3	1	b47DE
b47	c21	3	1	b47c21
c21	DIC	.	2	c21DIC
m23	b48	.	1	m23b48
b48	DE	3	1	b48DE
b48	c19	3	1	b48c19
m26	b49	.	1	m26b49
b49	DE	2	1	b49DE
b49	c16	2	1	b49c16
m26	b50	.	1	m26b50
b50	DE	2	1	b50DE
b50	c22	2	1	b50c22
c22	DIR	.	2	c22DIR
m26	b51	.	1	m26b51
b51	DE	2	1	b51DE
m26	b52	.	1	m26b52
b52	DE	2	1	b52DE
m27	b53	.	1	m27b53
b53	DE	2	1	b53DE
b53	c18	2	1	b53c18
m27	b54	.	1	m27b54
b54	DE	2	1	b54DE
b54	c23	2	1	b54c23
c23	DIR	.	2	c23DIR
m27	b55	.	1	m27b55
b55	DE	2	1	b55DE
m27	b56	.	1	m27b56

b56 DE 2 1 b56DE  
 b56 c22 2 1 b56c22  
 m28 b57 . 1 m28b57  
 b57 DE 3 1 b57DE  
 b57 c20 3 1 b57c20  
 m28 b58 . 1 m28b58  
 b58 DE 3 1 b58DE  
 b58 c24 3 1 b58c24  
 c24 DIR . 2 c24DIR  
 m28 b59 . 1 m28b59  
 b59 DE 3 1 b59DE  
 m28 b60 . 1 m28b60  
 b60 DE 3 1 b60DE  
 b60 c23 3 1 b60c23  
 m29 b61 . 1 m29b61  
 b61 DE 1 1 b61DE  
 b61 c21 1 1 b61c21  
 m29 b62 . 1 m29b62  
 b62 DE 1 1 b62DE  
 m29 b63 . 1 m29b63  
 b63 DE 1 1 b63DE  
 m29 b64 . 1 m29b64  
 b64 DE 1 1 b64DE  
 b64 c24 1 1 b64c24

;

data cond;

input

Sm8	Sm9	Sm10	Sm11	Sm14	Sm15	Sm16	Sm17	Sm20	Sm21
Sm22	Sm23	Sm26	Sm27	Sm28	Sm29	m8b1	b1DE	m8b2	b2DE
b2c1	c1DIR	m8b3	b3DE	b3c2	c2DIC	m8b4	b4DE	m9b5	b5DE
m9b6	b6DE	b6c3	c3DIR	m9b7	b7DE	b7c4	c4DIC	m9b8	b8DE
b8c1	m10b9	b9DE	m10b10	b10DE	b10c5	c5DIR	m10b11	b11DE	b11c6
c6DIC	m10b12	b12DE	b12c3	m11b13	b13DE	m11b14	b14DE	m11b15	b15DE
b15c7	c7DIC	m11b16	b16DE	b16c5	m14b17	b17DE	b17c2	m14b18	b18DE
b18c8	c8DIR	m14b19	b19DE	b19c9	c9DIC	m14b20	b20DE	m15b21	b21DE
b21c4	m15b22	b22DE	b22c10	c10DIR	m15b23	b23DE	b23c11	c11DIC	m15b24
b24DE	b24c8	m16b25	b25DE	b25c6	m16b26	b26DE	b26c12	c12DIR	m16b27
b27DE	b27c13	c13DIC	m16b28	b28DE	b28c10	m17b29	b29DE	b29c7	m17b30
b30DE	m17b31	b31DE	b31c14	c14DIC	m17b32	b32DE	b32c12	m20b33	b33DE
b33c9	m20b34	b34DE	b34c15	c15DIR	m20b35	b35DE	b35c16	c16DIC	m20b36
b36DE	m21b37	b37DE	b37c11	m21b38	b38DE	b38c17	c17DIR	m21b39	b39DE
b39c18	c18DIC	m21b40	b40DE	b40c15	m22b41	b41DE	b41c13	m22b42	b42DE
b42c19	c19DIR	m22b43	b43DE	b43c20	c20DIC	m22b44	b44DE	b44c17	m23b45
b45DE	b45c14	m23b46	b46DE	m23b47	b47DE	b47c21	c21DIC	m23b48	b48DE
b48c19	m26b49	b49DE	b49c16	m26b50	b50DE	b50c22	c22DIR	m26b51	b51DE
m26b52	b52DE	m27b53	b53DE	b53c18	m27b54	b54DE	b54c23	c23DIR	m27b55
b55DE	m27b56	b56DE	b56c22	m28b57	b57DE	b57c20	m28b58	b58DE	b58c24
c24DIR	m28b59	b59DE	m28b60	b60DE	b60c23	m29b61	b61DE	b61c21	m29b62
b62DE	m29b63	b63DE	m29b64	b64DE	b64c24	_type_\$	_rhs_;		

cards;

..... 1 .....

[illegible]

```

EQ 0
1
-1
EQ 0
1 -1
EQ 0
1 -1
EQ 0
1 -1
EQ 0
1 -1
EQ 0
1 -1
EQ 0
1 -1

```











```

. . . . .
. . . . .
. . . . . 1 . -1 . . -1 . . 1 . . . . .
. . . . .
. . . . .
. . . . . EQ 0

```

```

. . . . .
. . . . .
. . . . . 1 . 1 . -1 . . -1 . . . . .
. . . . .
. . . . .
. . . . . EQ 0

```

```

. . . . .
. . . . .
. . . . . 1 . -1 . 1 . . -1 . . . . .
. . . . .
. . . . .
. . . . . EQ 0

```

```

. . . . .
. . . . .
. . . . . 1 . -1 . -1 . . 1 . . . . .
. . . . .
. . . . .
. . . . . EQ 0

```

```

. . . . .
. . . . .
. . . . . 1 . -1 . . -1 . . . . .
. . . . .
. . . . .
. . . . . EQ 0

```

```

. . . . .
. . . . .
. . . . . 1 . . . . -1 . . . . .
. . . . .
. . . . .
. . . . . EQ 0

```

```

. . . . .
. . . . .
. . . . . 1 . . . . -1 . . . . .
. . . . .
. . . . .
. . . . . EQ 0

```

```

.....
.....
.....      1      1      -1      -1      .....
.....
.....      EQ 0

```

[illegible]

```
. . . . .
. . . . .
. . . . .      1   -1   -1   1   . . . . .
. . . . .
. . . . .      EQ = 0
```

```

.....
.....
.....
.....      1      1      -1      -1
.....
..... EQ 0

```

```

. . . . .
. . . . .
. . . . .
. . . . .      1      -1      1      -1
. . . . .
. . . . .      EQ 0

```

```
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .      1      -1      -1      1
. . . . .
. . . . .      EQ 0
```

[illegible]

. . . . . 1  
 . -1 . 1 . . -1 . . . . .  
 . . . . . EQ 0

. . . . . 1  
 . -1 . -1 . . 1 . . . . .  
 . . . . . EQ 0

. . . . .  
 . . . . . 1 . 1 . . -1 . -1 . . . . .  
 . . . . . EQ 0

. . . . .  
 . . . . . 1 . -1 . . 1 . -1 . . . . .  
 . . . . . EQ 0

. . . . .  
 . . . . . 1 . -1 . . -1 . 1 . . . . .  
 . . . . . EQ 0

. . . . .  
 . . . . . 1 . 1 . . -1 . -1 . . . . .  
 . . . . . EQ 0

. . . . .  
 . . . . . 1 . -1 . . 1 . -1 . . . . .  
 . . . . . EQ 0

. . . . .  
 . . . . . 1 . -1 . . -1 . 1 . . . . .



C.17 Partial .log and .lis Output Files for Problem CELLBORDER1, Run 2

NOTE: Number of nodes= 108 .  
 NOTE: Number of supply nodes= 1 .  
 NOTE: Number of demand nodes= 2 .  
 NOTE: Total supply= 8 , total demand= 8 .  
 NOTE: Number of arcs= 216 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 122 .  
 NOTE: Of these, 115 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 8 .  
 NOTE: Number of <= side constraints= 0 .  
 NOTE: Number of == side constraints= 72 .  
 NOTE: Number of >= side constraints= 0 .  
 NOTE: Number of arc and nonarc variable side constraint coefficients= 240 .  
 NOTE: Number of iterations, optimizing with constraints= 97 .  
 NOTE: Of these, 89 were degenerate.  
 NOTE: Optimum reached.  
 NOTE: Minimal total cost= 12 .  
 NOTE: The data set WORK.SOLUTION has 216 observations and 14 variables.  
 765        print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed page 1.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

_N	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
1	b25	DE	2	1	0	B25DE	.
2	b26	DE	2	1	0	B26DE	.
3	b28	DE	2	1	0	B28DE	.
4	b42	DE	1	1	0	B42DE	.
5	b43	DE	1	1	0	B43DE	.
6	b44	DE	1	1	0	B44DE	.
7	c13	DIC	0	2	0	C13DIC	.
8	m16	b25	0	1	0	M16B25	.
9	m16	b26	0	1	0	M16B26	.
10	m16	b27	0	1	0	M16B27	.
11	m16	b28	0	1	0	M16B28	.
12	m22	b41	0	1	0	M22B41	.
13	m22	b42	0	1	0	M22B42	.
14	m22	b43	0	1	0	M22B43	.
15	m22	b44	0	1	0	M22B44	.
16	b27	c13	2	1	0	B27C13	.
17	b41	c13	1	1	0	B41C13	.
18	S	m16	0	4	0	SM16	8
19	S	m22	0	4	0	SM22	8
_N_	_DEMAND_	_FLOW_	_FCOST_	_RCOST_	_STATUS_		
1	6	1	2	.	NONKEY ARC BASIC		



2	6	1	2	.	NONKEY ARC BASIC
3	6	1	2	.	NONKEY ARC BASIC
4	6	1	1	.	KEY_ARC BASIC
5	6	1	1	.	KEY_ARC BASIC
6	6	1	1	.	KEY_ARC BASIC
7	2	2	0	.	KEY_ARC BASIC
8	.	1	0	.	KEY_ARC BASIC
9	.	1	0	.	KEY_ARC BASIC
10	.	1	0	.	KEY_ARC BASIC
11	.	1	0	.	KEY_ARC BASIC
12	.	1	0	-2	UPPERBD NONBASIC
13	.	1	0	.	NONKEY ARC BASIC
14	.	1	0	.	NONKEY ARC BASIC
15	.	1	0	.	KEY_ARC BASIC
16	.	1	2	.	NONKEY ARC BASIC
17	.	1	1	.	KEY_ARC BASIC
18	.	4	0	.	KEY_ARC BASIC
19	.	4	0	.	KEY_ARC BASIC

# C.18 Partial .log and .lis Output Files for Problem CELLBORDER2A, Run 1

NOTE: Number of nodes= 108 .  
 NOTE: Number of supply nodes= 1 .  
 NOTE: Number of demand nodes= 3 .  
 NOTE: Total supply= 24 , total demand= 24 .  
 NOTE: Number of arcs= 216 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 140 .  
 NOTE: Of these, 117 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 24 .  
 NOTE: Number of <= side constraints= 0 .  
 NOTE: Number of == side constraints= 72 .  
 NOTE: Number of >= side constraints= 0 .  
 NOTE: Number of arc and nonarc variable side constraint coefficients= 240 .  
 NOTE: Number of iterations, optimizing with constraints= 123 .  
 NOTE: Of these, 105 were degenerate.  
 NOTE: Optimum reached.  
 NOTE: Minimal total cost= 32 .  
 NOTE: The data set WORK.SOLUTION has 216 observations and 14 variables.  
 765-       print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed pages 1-3.

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
1	b1	DE	0	1	0	B1DE	.
2	b4	DE	0	1	0	B4DE	.
3	b5	DE	0	1	0	B5DE	.
4	b6	DE	0	1	0	B6DE	.
5	b20	DE	0	1	0	B20DE	.
6	b22	DE	0	1	0	B22DE	.
7	b35	DE	0	1	0	B35DE	.
8	b36	DE	0	1	0	B36DE	.
9	b38	DE	0	1	0	B38DE	.
10	b39	DE	0	1	0	B39DE	.
11	c2	DIC	0	2	0	C2DIC	.
12	c4	DIC	0	2	0	C4DIC	.
13	c9	DIC	0	2	0	C9DIC	.
14	c11	DIC	0	2	0	C11DIC	.
15	c1	DIR	0	2	0	C1DIR	.
16	c8	DIR	0	2	0	C8DIR	.
17	c15	DIR	0	2	0	C15DIR	.
18	m8	b1	0	1	0	M8B1	.
19	m14	b17	0	1	0	M14B17	.
20	m14	b18	0	1	0	M14B18	.
21	m14	b19	0	1	0	M14B19	.
22	m8	b2	0	1	0	M8B2	.
23	m14	b20	0	1	0	M14B20	.
24	m15	b21	0	1	0	M15B21	.
25	m15	b22	0	1	0	M15B22	.
26	m15	b23	0	1	0	M15B23	.

27	m15	b24	0	1	0	M15B24	.
28	m8	b3	0	1	0	M8B3	.
29	m20	b33	0	1	0	M20B33	.
30	m20	b34	0	1	0	M20B34	.
31	m20	b35	0	1	0	M20B35	.
32	m20	b36	0	1	0	M20B36	.
33	m21	b37	0	1	0	M21B37	.
34	m21	b38	0	1	0	M21B38	.
35	m21	b39	0	1	0	M21B39	.
36	m8	b4	0	1	0	M8B4	.
37	m21	b40	0	1	0	M21B40	.
38	m9	b5	0	1	0	M9B5	.
39	m9	b6	0	1	0	M9B6	.
40	m9	b7	0	1	0	M9B7	.
41	m9	b8	0	1	0	M9B8	.
42	b2	c1	0	1	0	B2C1	.
43	b8	c1	0	1	0	B8C1	.
44	b23	c11	0	1	0	B23C11	.
45	b37	c11	0	1	0	B37C11	.
46	b34	c15	0	1	0	B34C15	.
47	b40	c15	0	1	0	B40C15	.
48	b3	c2	0	1	0	B3C2	.
49	b17	c2	0	1	0	B17C2	.
50	b7	c4	0	1	0	B7C4	.
51	b21	c4	0	1	0	B21C4	.
52	b18	c8	0	1	0	B18C8	.
53	b24	c8	0	1	0	B24C8	.
54	b19	c9	0	1	0	B19C9	.
55	b33	c9	0	1	0	B33C9	.
56	S	m14	1	4	0	SM14	24
57	S	m15	1	4	0	SM15	24
58	S	m20	2	4	0	SM20	24
59	S	m21	2	4	0	SM21	24
60	S	m8	1	4	0	SM8	24
61	S	m9	1	4	0	SM9	24

_N_	_DEMAND_	_FLOW_	_FCOST_	_RCOST_	_STATUS_
1	10	1	0	.	KEY_ARC BASIC
2	10	1	0	.	KEY_ARC BASIC
3	10	1	0	.	KEY_ARC BASIC
4	10	1	0	.	KEY_ARC BASIC
5	10	1	0	.	NONKEY ARC BASIC
6	10	1	0	.	NONKEY ARC BASIC
7	10	1	0	.	NONKEY ARC BASIC
8	10	1	0	.	NONKEY ARC BASIC
9	10	1	0	0	UPPERBD NONBASIC
10	10	1	0	.	NONKEY ARC BASIC
11	8	2	0	.	KEY_ARC BASIC
12	8	2	0	.	KEY_ARC BASIC
13	8	2	0	.	KEY_ARC BASIC

14	8	2	0	.	KEY_ARC	BASIC
15	6	2	0	.	KEY_ARC	BASIC
16	6	2	0	.	KEY_ARC	BASIC
17	6	2	0	.	NONKEY ARC	BASIC
18	.	1	0	.	NONKEY ARC	BASIC
19	.	1	0	.	NONKEY ARC	BASIC
20	.	1	0	.	NONKEY ARC	BASIC
21	.	1	0	.	NONKEY ARC	BASIC
22	.	1	0	.	NONKEY ARC	BASIC
23	.	1	0	.	KEY_ARC	BASIC
24	.	1	0	.	NONKEY ARC	BASIC
25	.	1	0	.	KEY_ARC	BASIC
26	.	1	0	.	KEY_ARC	BASIC
27	.	1	0	.	KEY_ARC	BASIC
28	.	1	0	.	KEY_ARC	BASIC
29	.	1	0	.	KEY_ARC	BASIC
30	.	1	0	.	KEY_ARC	BASIC
31	.	1	0	.	KEY_ARC	BASIC
32	.	1	0	.	KEY_ARC	BASIC
33	.	1	0	.	KEY_ARC	BASIC
34	.	1	0	.	KEY_ARC	BASIC
35	.	1	0	.	KEY_ARC	BASIC
36	.	1	0	.	NONKEY ARC	BASIC
37	.	1	0	.	KEY_ARC	BASIC
38	.	1	0	.	NONKEY ARC	BASIC
39	.	1	0	.	NONKEY ARC	BASIC
40	.	1	0	.	KEY_ARC	BASIC
41	.	1	0	.	NONKEY ARC	BASIC
42	.	1	0	.	KEY_ARC	BASIC
43	.	1	0	.	NONKEY ARC	BASIC
44	.	1	0	.	NONKEY ARC	BASIC
45	.	1	0	.	NONKEY ARC	BASIC
46	.	1	0	.	KEY_ARC	BASIC
47	.	1	0	.	NONKEY ARC	BASIC
48	.	1	0	.	NONKEY ARC	BASIC
49	.	1	0	.	KEY_ARC	BASIC
50	.	1	0	.	NONKEY ARC	BASIC
51	.	1	0	.	KEY_ARC	BASIC
52	.	1	0	.	KEY_ARC	BASIC
53	.	1	0	.	NONKEY ARC	BASIC
54	.	1	0	.	NONKEY ARC	BASIC
55	.	1	0	.	NONKEY ARC	BASIC
56	.	4	4	.	KEY_ARC	BASIC
57	.	4	4	.	KEY_ARC	BASIC
58	.	4	8	.	KEY_ARC	BASIC
59	.	4	8	.	KEY_ARC	BASIC
60	.	4	4	.	KEY_ARC	BASIC
61	.	4	4	.	KEY_ARC	BASIC

### C.19 Partial .log and .lis Output Files for Problem CELLBORDER2B, Run 1

NOTE: Number of nodes= 108 .  
 NOTE: Number of supply nodes= 1 .  
 NOTE: Number of demand nodes= 3 .  
 NOTE: Total supply= 24 , total demand= 24 .  
 NOTE: Number of arcs= 216 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 139 .  
 NOTE: Of these, 116 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 24 .  
 NOTE: Number of <= side constraints= 0 .  
 NOTE: Number of == side constraints= 72 .  
 NOTE: Number of >= side constraints= 0 .  
 NOTE: Number of arc and nonarc variable side constraint coefficients= 240 .  
 NOTE: Number of iterations, optimizing with constraints= 130 .  
 NOTE: Of these, 107 were degenerate.  
 NOTE: Optimum reached.  
 NOTE: Minimal total cost= 26 .  
 NOTE: The data set WORK.SOLUTION has 216 observations and 14 variables.  
 765        print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed pages 1-4.

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
99	S	m14	1	4	0	SM14	24
100	S	m15	1	4	0	SM15	24
101	S	m16	1	4	0	SM16	24
102	S	m17	1	4	0	SM17	24
103	S	m20	2	4	0	SM20	24
104	S	m21	2	4	0	SM21	24
105	S	m22	1	4	0	SM22	24
106	S	m23	1	4	0	SM23	24
107	S	m8	1	4	0	SM8	24
108	S	m9	1	4	0	SM9	24

_N_	_DEMAND_	_FLOW_	_FCOST_	_RCOST_	_STATUS_
99	.	3	3	.	KEY_ARC BASIC
100	.	3	3	.	KEY_ARC BASIC
101	.	3	3	.	KEY_ARC BASIC
102	.	3	3	.	KEY_ARC BASIC
103	.	1	2	.	KEY_ARC BASIC
104	.	1	2	.	KEY_ARC BASIC
105	.	2	2	.	KEY_ARC BASIC
106	.	2	2	.	KEY_ARC BASIC
107	.	3	3	.	KEY_ARC BASIC
108	.	3	3	.	KEY_ARC BASIC

*C:20 Partial .log and .lis Output Files for Problem CELLBORDER2B, Run 2*

NOTE: Number of nodes= 108 .  
 NOTE: Number of supply nodes= 1 .  
 NOTE: Number of demand nodes= 3 .  
 NOTE: Total supply= 24 , total demand= 24 .  
 NOTE: Number of arcs= 216 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 139 .  
 NOTE: Of these, 116 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 24 .  
 NOTE: Number of <= side constraints= 0 .  
 NOTE: Number of == side constraints= 73 .  
 NOTE: Number of >= side constraints= 0 .  
 NOTE: Number of arc and nonarc variable side constraint coefficients= 241 .  
 NOTE: Number of iterations, optimizing with constraints= 113 .  
 NOTE: Of these, 98 were degenerate.  
 NOTE: Optimum reached.  
 NOTE: Minimal total cost= 28 .  
 NOTE: The data set WORK.SOLUTION has 216 observations and 14 variables.  
 772            print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed pages 1-4.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_		
100	S	m14	1	4	0	SM14	24		
101	S	m15	1	4	0	SM15	24		
102	S	m16	1	4	0	SM16	24		
103	S	m17	1	4	0	SM17	24		
104	S	m20	2	4	0	SM20	24		
105	S	m21	2	4	0	SM21	24		
106	S	m22	1	4	0	SM22	24		
107	S	m23	1	4	0	SM23	24		
108	S	m8	1	4	0	SM8	24		
109	S	m9	1	4	0	SM9	24		
_N_			_DEMAND_	_FLOW_	_FCOST_	_RCOST_	_STATUS_		
100		.		4	4	.	KEY_ARC	BASIC	
101		.	2.666666667		2.666666667	.	KEY_ARC	BASIC	
102		.	2.666666667		2.666666667	.	KEY_ARC	BASIC	
103		.	2.666666667		2.666666667	.	KEY_ARC	BASIC	
104		.		2	4	.	KEY_ARC	BASIC	
105		.		2	4	.	KEY_ARC	BASIC	
106		.	2.666666667		2.666666667	.	KEY_ARC	BASIC	
107		.	2.666666667		2.666666667	.	KEY_ARC	BASIC	
108		.	1.333333333		1.333333333	.	KEY_ARC	BASIC	
109		.	1.333333333		1.333333333	.	KEY_ARC	BASIC	

## C.21 Partial .log and .lis Output Files for Problem CELLBORDER2B, Run 3

NOTE: Number of nodes= 108 .  
 NOTE: Number of supply nodes= 1 .  
 NOTE: Number of demand nodes= 3 .  
 NOTE: Total supply= 24 , total demand= 24 .  
 NOTE: Number of arcs= 216 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 136 .  
 NOTE: Of these, 112 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 24 .  
 NOTE: Number of <= side constraints= 0 .  
 NOTE: Number of == side constraints= 72 .  
 NOTE: Number of >= side constraints= 16 .  
 NOTE: Number of arc and nonarc variable side constraint coefficients= 304 .  
 NOTE: Number of iterations, optimizing with constraints= 158 .  
 NOTE: Of these, 138 were degenerate.  
 NOTE: Optimum reached.  
 NOTE: Minimal total cost= 26 .  
 NOTE: The data set WORK.SOLUTION has 216 observations and 14 variables.  
 877            print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed pages 1-4.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
20	c2	DIC	0	2	0	C2DIC	.
21	c4	DIC	0	2	0	C4DIC	.
22	c9	DIC	0	2	0	C9DIC	.
23	c11	DIC	0	2	0	C11DIC	.
24	c13	DIC	0	2	0	C13DIC	.
25	c14	DIC	0	2	0	C14DIC	.
26	c1	DIR	0	2	0	C1DIR	.
27	c8	DIR	0	2	0	C8DIR	.
28	c10	DIR	0	2	0	C10DIR	.
29	c12	DIR	0	2	0	C12DIR	.
30	c15	DIR	0	2	0	C15DIR	.
31	c17	DIR	0	2	0	C17DIR	.
32	c19	DIR	0	2	0	C19DIR	.
99	S	m14	0	4	0	SM14	24
100	S	m15	0	4	0	SM15	24
101	S	m16	0	4	0	SM16	24
102	S	m17	0	4	0	SM17	24
103	S	m20	0	4	0	SM20	24
104	S	m21	0	4	0	SM21	24
105	S	m22	0	4	0	SM22	24
106	S	m23	0	4	0	SM23	24
107	S	m8	0	4	0	SM8	24

108	S	m9	0	4	0	SM9	24
<u>_N_</u>	<u>_DEMAND_</u>	<u>_FLOW_</u>	<u>_FCOST_</u>	<u>_RCOST_</u>	<u>_STATUS_</u>		
20	6	1	0	.	KEY_ARC	BASIC	
21	6	1	0	.	KEY_ARC	BASIC	
22	6	0.5	0	.	KEY_ARC	BASIC	
23	6	0.5	0	.	KEY_ARC	BASIC	
24	6	1.5	0	.	KEY_ARC	BASIC	
25	6	1.5	0	.	NONKEY ARC	BASIC	
26	8	1	0	.	KEY_ARC	BASIC	
27	8	1.5	0	.	KEY_ARC	BASIC	
28	8	1.5	0	.	KEY_ARC	BASIC	
29	8	1.5	0	.	KEY_ARC	BASIC	
30	8	0.5	0	.	NONKEY ARC	BASIC	
31	8	0.5	0	.	KEY_ARC	BASIC	
32	8	1.5	0	.	KEY_ARC	BASIC	
99	.	3	0	.	KEY_ARC	BASIC	
100	.	3	0	.	KEY_ARC	BASIC	
101	.	3	0	.	KEY_ARC	BASIC	
102	.	3	0	.	KEY_ARC	BASIC	
103	.	1	0	.	KEY_ARC	BASIC	
104	.	1	0	.	KEY_ARC	BASIC	
105	.	3	0	.	KEY_ARC	BASIC	
106	.	3	0	.	KEY_ARC	BASIC	
107	.	2	0	.	KEY_ARC	BASIC	
108	.	2	0	.	KEY_ARC	BASIC	



C:22 Partial .log and .lis Output Files for Problem CELLBORDER2B, Run 5

NOTE: Number of nodes= 108 .  
 NOTE: Number of supply nodes= 1 .  
 NOTE: Number of demand nodes= 3 .  
 NOTE: Total supply= 24 , total demand= 24 .  
 NOTE: Number of arcs= 216 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 136 .  
 NOTE: Of these, 112 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 24 .  
 NOTE: Number of <= side constraints= 0 .  
 NOTE: Number of == side constraints= 72 .  
 NOTE: Number of >= side constraints= 32 .  
 NOTE: Number of arc and nonarc variable side constraint coefficients= 368 .  
 NOTE: Number of iterations, optimizing with constraints= 179 .  
 NOTE: Of these, 145 were degenerate.  
 NOTE: Optimum reached.  
 NOTE: Minimal total cost= 26 .  
 NOTE: The data set WORK.SOLUTION has 216 observations and 14 variables.  
 989        print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed pages 1-4.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
20	c2	DIC	0	2	0	C2DIC	.
21	c4	DIC	0	2	0	C4DIC	.
22	c9	DIC	0	2	0	C9DIC	.
23	c11	DIC	0	2	0	C11DIC	.
24	c13	DIC	0	2	0	C13DIC	.
25	c14	DIC	0	2	0	C14DIC	.
26	c1	DIR	0	2	0	C1DIR	.
27	c8	DIR	0	2	0	C8DIR	.
28	c10	DIR	0	2	0	C10DIR	.
29	c12	DIR	0	2	0	C12DIR	.
30	c15	DIR	0	2	0	C15DIR	.
31	c17	DIR	0	2	0	C17DIR	.
32	c19	DIR	0	2	0	C19DIR	.
99	S	m14	0	4	0	SM14	24
100	S	m15	0	4	0	SM15	24
101	S	m16	0	4	0	SM16	24
102	S	m17	0	4	0	SM17	24
103	S	m20	0	4	0	SM20	24
104	S	m21	0	4	0	SM21	24
105	S	m22	0	4	0	SM22	24
106	S	m23	0	4	0	SM23	24
107	S	m8	0	4	0	SM8	24

108	S	m9	0	4	0	SM9	24
<u>_N_</u>	<u>_DEMAND_</u>	<u>_FLOW_</u>	<u>_FCOST_</u>	<u>_RCOST_</u>	<u>_STATUS_</u>		
20	6	1	0	.	KEY_ARC	BASIC	
21	6	1	0	.	KEY_ARC	BASIC	
22	6	0.5	0	.	KEY_ARC	BASIC	
23	6	0.5	0	.	KEY_ARC	BASIC	
24	6	1.5	0	.	KEY_ARC	BASIC	
25	6	1.5	0	.	NONKEY ARC	BASIC	
26	8	1	0	.	KEY_ARC	BASIC	
27	8	1.5	0	.	KEY_ARC	BASIC	
28	8	1.5	0	.	KEY_ARC	BASIC	
29	8	1.5	0	.	NONKEY ARC	BASIC	
30	8	0.5	0	.	NONKEY ARC	BASIC	
31	8	0.5	0	.	KEY_ARC	BASIC	
32	8	1.5	0	.	KEY_ARC	BASIC	
99	.	3	0	.	KEY_ARC	BASIC	
100	.	3	0	.	KEY_ARC	BASIC	
101	.	3	0	.	KEY_ARC	BASIC	
102	.	3	0	.	KEY_ARC	BASIC	
103	.	1	0	.	KEY_ARC	BASIC	
104	.	1	0	.	KEY_ARC	BASIC	
105	.	3	0	.	KEY_ARC	BASIC	
106	.	3	0	.	KEY_ARC	BASIC	
107	.	2	0	.	KEY_ARC	BASIC	
108	.	2	0	.	KEY_ARC	BASIC	

### C.23 Input File for Problem CELLBORDER2B, Run 6

```
data noded;
  input _node_$ _sd_;
  cards;
S    64
DE   -10
DIR   -8
DIC   -6
DN   -40
;
data arcd;
input _from_$ _to_$ _cost_ _capac_ _name_$;
cards;
S m8 . 4 Sm8
S m9 . 4 Sm9
S m10 . 4 Sm10
S m11 . 4 Sm11
S m14 . 4 Sm14
S m15 . 4 Sm15
S m16 . 4 Sm16
S m17 . 4 Sm17
S m20 . 4 Sm20
S m21 . 4 Sm21
S m22 . 4 Sm22
S m23 . 4 Sm23
S m26 . 4 Sm26
S m27 . 4 Sm27
S m28 . 4 Sm28
S m29 . 4 Sm29
m8 b1 . 1 m8b1
b1 DE 1 1 b1DE
b1 DN . 1 b1DN
m8 b2 . 1 m8b2
b2 DE 1 1 b2DE
b2 DN . 1 b2DN
b2 c1 1 1 b2c1
c1 DIR . 2 c1DIR
m8 b3 . 1 m8b3
b3 DE 1 1 b3DE
b3 DN . 1 b3DN
b3 c2 1 1 b3c2
c2 DIC . 2 c2DIC
m8 b4 . 1 m8b4
b4 DE 1 1 b4DE
b4 DN . 1 b4DN
m9 b5 . 1 m9b5
b5 DE 1 1 b5DE
b5 DN . 1 b5DN
m9 b6 . 1 m9b6
b6 DE 1 1 b6DE
```

b6 DN . 1 b6DN  
 b6 c3 1 1 b6c3  
 c3 DIR . 2 c3DIR  
 m9 b7 . 1 m9b7  
 b7 DE 1 1 b7DE  
 b7 DN . 1 b7DN  
 b7 c4 1 1 b7c4  
 c4 DIC . 2 c4DIC  
 m9 b8 . 1 m9b8  
 b8 DE 1 1 b8DE  
 b8 DN . 1 b8DN  
 b8 c1 1 1 b8c1  
 m10 b9 . 1 m10b9  
 b9 DE 7 1 b9DE  
 b9 DN . 1 b9DN  
 m10 b10 . 1 m10b10  
 b10 DE 7 1 b10DE  
 b10 DN . 1 b10DN  
 b10 c5 7 1 b10c5  
 c5 DIR . 2 c5DIR  
 m10 b11 . 1 m10b11  
 b11 DE 7 1 b11DE  
 b11 DN . 1 b11DN  
 b11 c6 7 1 b11c6  
 c6 DIC . 2 c6DIC  
 m10 b12 . 1 m10b12  
 b12 DE 7 1 b12DE  
 b12 DN . 1 b12DN  
 b12 c3 7 1 b12c3  
 m11 b13 . 1 m11b13  
 b13 DE 7 1 b13DE  
 b13 DN . 1 b13DN  
 m11 b14 . 1 m11b14  
 b14 DE 7 1 b14DE  
 b14 DN . 1 b14DN  
 m11 b15 . 1 m11b15  
 b15 DE 7 1 b15DE  
 b15 DN . 1 b15DN  
 b15 c7 7 1 b15c7  
 c7 DIC . 2 c7DIC  
 m11 b16 . 1 m11b16  
 b16 DE 7 1 b16DE  
 b16 DN . 1 b16DN  
 b16 c5 7 1 b16c5  
 m14 b17 . 1 m14b17  
 b17 DE 1 1 b17DE  
 b17 DN . 1 b17DN  
 b17 c2 1 1 b17c2  
 m14 b18 . 1 m14b18  
 b18 DE 1 1 b18DE  
 b18 DN . 1 b18DN

b18 c8 1 1 b18c8  
 c8 DIR . 2 c8DIR  
 m14 b19 . 1 m14b19  
 b19 DE 1 1 b19DE  
 b19 DN . 1 b19DN  
 b19 c9 1 1 b19c9  
 c9 DIC . 2 c9DIC  
 m14 b20 . 1 m14b20  
 b20 DE 1 1 b20DE  
 b20 DN . 1 b20DN  
 m15 b21 . 1 m15b21  
 b21 DE 1 1 b21DE  
 b21 DN . 1 b21DN  
 b21 c4 1 1 b21c4  
 m15 b22 . 1 m15b22  
 b22 DE 1 1 b22DE  
 b22 DN . 1 b22DN  
 b22 c10 1 1 b22c10  
 c10 DIR . 2 c10DIR  
 m15 b23 . 1 m15b23  
 b23 DE 1 1 b23DE  
 b23 DN . 1 b23DN  
 b23 c11 1 1 b23c11  
 c11 DIC . 2 c11DIC  
 m15 b24 . 1 m15b24  
 b24 DE 1 1 b24DE  
 b24 DN . 1 b24DN  
 b24 c8 1 1 b24c8  
 m16 b25 . 1 m16b25  
 b25 DE 1 1 b25DE  
 b25 DN . 1 b25DN  
 b25 c6 1 1 b25c6  
 m16 b26 . 1 m16b26  
 b26 DE 1 1 b26DE  
 b26 DN . 1 b26DN  
 b26 c12 1 1 b26c12  
 c12 DIR . 2 c12DIR  
 m16 b27 . 1 m16b27  
 b27 DE 1 1 b27DE  
 b27 DN . 1 b27DN  
 b27 c13 1 1 b27c13  
 c13 DIC . 2 c13DIC  
 m16 b28 . 1 m16b28  
 b28 DE 1 1 b28DE  
 b28 DN . 1 b28DN  
 b28 c10 1 1 b28c10  
 m17 b29 . 1 m17b29  
 b29 DE 1 1 b29DE  
 b29 DN . 1 b29DN  
 b29 c7 1 1 b29c7  
 m17 b30 . 1 m17b30

b30 DE 1 1 b30DE  
 b30 DN . 1 b30DN  
 m17 b31 . 1 m17b31  
 b31 DE 1 1 b31DE  
 b31 DN . 1 b31DN  
 b31 c14 1 1 b31c14  
 c14 DIC . 2 c14DIC  
 m17 b32 . 1 m17b32  
 b32 DE 1 1 b32DE  
 b32 DN . 1 b32DN  
 b32 c12 1 1 b32c12  
 m20 b33 . 1 m20b33  
 b33 DE 2 1 b33DE  
 b33 DN . 1 b33DN  
 b33 c9 2 1 b33c9  
 m20 b34 . 1 m20b34  
 b34 DE 2 1 b34DE  
 b34 DN . 1 b34DN  
 b34 c15 2 1 b34c15  
 c15 DIR . 2 c15DIR  
 m20 b35 . 1 m20b35  
 b35 DE 2 1 b35DE  
 b35 DN . 1 b35DN  
 b35 c16 2 1 b35c16  
 c16 DIC . 2 c16DIC  
 m20 b36 . 1 m20b36  
 b36 DE 2 1 b36DE  
 b36 DN . 1 b36DN  
 m21 b37 . 1 m21b37  
 b37 DE 2 1 b37DE  
 b37 DN . 1 b37DN  
 b37 c11 2 1 b37c11  
 m21 b38 . 1 m21b38  
 b38 DE 2 1 b38DE  
 b38 DN . 1 b38DN  
 b38 c17 2 1 b38c17  
 c17 DIR . 2 c17DIR  
 m21 b39 . 1 m21b39  
 b39 DE 2 1 b39DE  
 b39 DN . 1 b39DN  
 b39 c18 2 1 b39c18  
 c18 DIC . 2 c18DIC  
 m21 b40 . 1 m21b40  
 b40 DE 2 1 b40DE  
 b40 DN . 1 b40DN  
 b40 c15 2 1 b40c15  
 m22 b41 . 1 m22b41  
 b41 DE 1 1 b41DE  
 b41 DN . 1 b41DN  
 b41 c13 1 1 b41c13  
 m22 b42 . 1 m22b42

b42 DE 1 1 b42DE  
 b42 DN . 1 b42DN  
 b42 c19 1 1 b42c19  
 c19 DIR . 2 c19DIR  
 m22 b43 . 1 m22b43  
 b43 DE 1 1 b43DE  
 b43 DN . 1 b43DN  
 b43 c20 1 1 b43c20  
 c20 DIC . 2 c20DIC  
 m22 b44 . 1 m22b44  
 b44 DE 1 1 b44DE  
 b44 DN . 1 b44DN  
 b44 c17 1 1 b44c17  
 m23 b45 . 1 m23b45  
 b45 DE 1 1 b45DE  
 b45 DN . 1 b45DN  
 b45 c14 1 1 b45c14  
 m23 b46 . 1 m23b46  
 b46 DE 1 1 b46DE  
 b46 DN . 1 b46DN  
 m23 b47 . 1 m23b47  
 b47 DE 1 1 b47DE  
 b47 DN . 1 b47DN  
 b47 c21 1 1 b47c21  
 c21 DIC . 2 c21DIC  
 m23 b48 . 1 m23b48  
 b48 DE 1 1 b48DE  
 b48 DN . 1 b48DN  
 b48 c19 1 1 b48c19  
 m26 b49 . 1 m26b49  
 b49 DE 7 1 b49DE  
 b49 DN . 1 b49DN  
 b49 c16 7 1 b49c16  
 m26 b50 . 1 m26b50  
 b50 DE 7 1 b50DE  
 b50 DN . 1 b50DN  
 b50 c22 7 1 b50c22  
 c22 DIR . 2 c22DIR  
 m26 b51 . 1 m26b51  
 b51 DE 7 1 b51DE  
 b51 DN . 1 b51DN  
 m26 b52 . 1 m26b52  
 b52 DE 7 1 b52DE  
 b52 DN . 1 b52DN  
 m27 b53 . 1 m27b53  
 b53 DE 2 1 b53DE  
 b53 DN . 1 b53DN  
 b53 c18 2 1 b53c18  
 m27 b54 . 1 m27b54  
 b54 DE 2 1 b54DE  
 b54 DN . 1 b54DN

b54 c23 2 1 b54c23  
 c23 DIR . 2 c23DIR  
 m27 b55 . 1 m27b55  
 b55 DE 2 1 b55DE  
 b55 DN . 1 b55DN  
 m27 b56 . 1 m27b56  
 b56 DE 2 1 b56DE  
 b56 DN . 1 b56DN  
 b56 c22 2 1 b56c22  
 m28 b57 . 1 m28b57  
 b57 DE 3 1 b57DE  
 b57 DN . 1 b57DN  
 b57 c20 3 1 b57c20  
 m28 b58 . 1 m28b58  
 b58 DE 3 1 b58DE  
 b58 DN . 1 b58DN  
 b58 c24 3 1 b58c24  
 c24 DIR . 2 c24DIR  
 m28 b59 . 1 m28b59  
 b59 DE 3 1 b59DE  
 b59 DN . 1 b59DN  
 m28 b60 . 1 m28b60  
 b60 DE 3 1 b60DE  
 b60 DN . 1 b60DN  
 b60 c23 3 1 b60c23  
 m29 b61 . 1 m29b61  
 b61 DE 3 1 b61DE  
 b61 DN . 1 b61DN  
 b61 c21 3 1 b61c21  
 m29 b62 . 1 m29b62  
 b62 DE 3 1 b62DE  
 b62 DN . 1 b62DN  
 m29 b63 . 1 m29b63  
 b63 DE 3 1 b63DE  
 b63 DN . 1 b63DN  
 m29 b64 . 1 m29b64  
 b64 DE 3 1 b64DE  
 b64 DN . 1 b64DN  
 b64 c24 3 1 b64c24

;

data cond;

input

Sm8	Sm9	Sm10	Sm11	Sm14	Sm15	Sm16	Sm17	Sm20	Sm21
Sm22	Sm23	Sm26	Sm27	Sm28	Sm29	m8b1	b1DE	b1DN	m8b2
b2DE	b2DN	b2c1	c1DIR	m8b3	b3DE	b3DN	b3c2	c2DIC	m8b4
b4DE	b4DN	m9b5	b5DE	b5DN	m9b6	b6DE	b6DN	b6c3	c3DIR
m9b7	b7DE	b7DN	b7c4	c4DIC	m9b8	b8DE	b8DN	b8c1	m10b9
b9DE	b9DN	m10b10	b10DE	b10DN	b10c5	c5DIR	m10b11	b11DE	b11DN
b11c6	c6DIC	m10b12	b12DE	b12DN	b12c3	m11b13	b13DE	b13DN	m11b14
b14DE	b14DN	m11b15	b15DE	b15DN	b15c7	c7DIC	m11b16	b16DE	b16DN
b16c5	m14b17	b17DE	b17DN	b17c2	m14b18	b18DE	b18DN	b18c8	c8DIR





EQ 0

-1

1

EQ 0

1

-1

EQ 0

1

-1

EQ 0

1

-1

EQ 0

1

-1

EQ 0



EQ 0

1

-1

EQ 0

-1

1

EQ 0

1

-1

EQ 0

1

-1

EQ 0

1

-1

EQ 0

. . . . .  
 . . . . .  
 . . . . . 1 . . . . .  
 . . . . . -1 . . . . .  
 EQ 0

. . . . .  
 . . . . .  
 . . . . .  
 . . . . .  
 . . . . . 1 . . . . .  
 . . . . . -1 . . . . .  
 EQ 0

. . . . .  
 . . . . .  
 . . . . .  
 . . . . .  
 . . . . .  
 . . . . .  
 . 1 . . . . . -1 . . . . .  
 EQ 0

. . . . .  
 . . . . .  
 . . . . .  
 . . . . .  
 . . . . .  
 . . . . .  
 . . . . . 1 . . . . . -1  
 EQ 0

. . . . . 1 . . 1 . . . -1 . . . -1 . . .  
 . . . . .  
 . . . . .  
 . . . . .  
 . . . . .  
 . . . . .  
 EQ 0

. . . . . 1 . . -1 . . . 1 . . . -1 . . .  
 . . . . .  
 . . . . .  
 . . . . .  
 . . . . .  
 . . . . .  
 EQ 0

```

. . . . . 1 1 .
. -1 . . -1 . . . . . 1 1 .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
EQ 0

```

[illegible]

```

. . . . . 1 . -1 .
. -1 . . . 1 . . . . . 1 . -1 .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
EQ 0

```

```

. . . . . 1 . 1 . . . -1 . . . -1 . . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
EQ 0.

```

```

. . . . .
. . . . . 1 . -1 . . . 1 . . . -1 . . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .

```

EQ 0

1 -1 -1 1

EQ 0

1 1 -1 -1

EQ 0

1 -1 1 -1

EQ 0

1 -1 -1 1

EQ 0

1 1 -1 -1

EQ 0

1 -1 1 -1

EQ 0

EQ 0

EQ 0

EQ 0

EQ 0

EQ 0





EQ 0

1 -1 1 -1

EQ 0

-1 1 1 -1

EQ 0

1 1 -1 -1

EQ 0

1 -1 1 -1

EQ 0

1 -1 -1 1

EQ 0

```

. . . . . 1 . . 1 . . -1 . .
-1 . . . . .

```

EQ 0

```

. . . . . 1 . . -1 . . 1 . .
-1 . . . . .

```

EQ 0

```

. . . . . 1 . . -1 . . -1 . .
1 . . . . .

```

EQ 0

```

. . . . . 1 . . 1 . . -1 . . -1 . .
. . . . .

```

EQ 0

```

. . . . . 1 . . -1 . . 1 . . -1 . .
. . . . .

```

EQ 0

```

. . . . . 1 . . -1 . . -1 . . 1 . .
. . . . .

```

EQ 0



-1 . . . -1 . . 1 . . . . .  
EQ 0

. . . . . 1 . . 1 . . -1 . -1 . . . . .  
EQ 0

. . . . . 1 . . -1 . . 1 . -1 . . . . .  
EQ 0

. . . . . 1 . . -1 . . -1 . 1 . . . . .  
EQ 0

. . . . . 1 . . 1 . -1 . -1 . . . . .  
EQ 0

. . . . . 1 . . -1 . 1 . -1 . . . . .  
EQ 0

EQ 0

```
run;
```

*C.24 Partial .log and .lis Output Files for Problem CELLBORDER2B, Run c*

NOTE: Number of nodes= 109 .  
 NOTE: Number of supply nodes= 1 .  
 NOTE: Number of demand nodes= 4 .  
 NOTE: Total supply= 64 , total demand= 64 .  
 NOTE: Number of arcs= 280 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 218 .  
 NOTE: Of these, 152 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 24 .  
 NOTE: Number of <= side constraints= 0 .  
 NOTE: Number of == side constraints= 72 .  
 NOTE: Number of >= side constraints= 0 .  
 NOTE: Number of arc and nonarc variable side constraint coefficients= 240 .  
 NOTE: Number of iterations, optimizing with constraints= 225 .  
 NOTE: Of these, 207 were degenerate.  
 NOTE: Optimum reached.  
 NOTE: Minimal total cost= 26 .  
 NOTE: The data set WORK.SOLUTION has 280 observations and 14 variables.  
 981        print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed pages 1-8.

NOTE: SAS Institute Inc., SAS Circle, PO Box 8000, Cary, NC 27512-8000

_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
1	b1	DE	1	1	0	B1DE	.
2	b4	DE	1	1	0	B4DE	.
3	b5	DE	1	1	0	B5DE	.
4	b6	DE	1	1	0	B6DE	.
5	b17	DE	1	1	0	B17DE	.
6	b19	DE	1	1	0	B19DE	.
7	b20	DE	1	1	0	B20DE	.
8	b21	DE	1	1	0	B21DE	.
9	b23	DE	1	1	0	B23DE	.
10	b25	DE	1	1	0	B25DE	.
11	b29	DE	1	1	0	B29DE	.
12	b30	DE	1	1	0	B30DE	.
13	b35	DE	2	1	0	B35DE	.
14	b36	DE	2	1	0	B36DE	.
15	b39	DE	2	1	0	B39DE	.
16	b43	DE	1	1	0	B43DE	.
17	b44	DE	1	1	0	B44DE	.
18	b46	DE	1	1	0	B46DE	.
19	b47	DE	1	1	0	B47DE	.
20	c2	DIC	0	2	0	C2DIC	.
21	c4	DIC	0	2	0	C4DIC	.
22	c9	DIC	0	2	0	C9DIC	.
23	c11	DIC	0	2	0	C11DIC	.

24	c13	DIC	0	2	0	C13DIC	.
25	c14	DIC	0	2	0	C14DIC	.
26	c1	DIR	0	2	0	C1DIR	.
27	c8	DIR	0	2	0	C8DIR	.
28	c10	DIR	0	2	0	C10DIR	.
29	c12	DIR	0	2	0	C12DIR	.
30	c15	DIR	0	2	0	C15DIR	.
31	c17	DIR	0	2	0	C17DIR	.
32	c19	DIR	0	2	0	C19DIR	.

161	b2	c1	1	1	0	B2C1	.
162	b8	c1	1	1	0	B8C1	.
163	b22	c10	1	1	0	B22C10	.
164	b28	c10	1	1	0	B28C10	.
165	b23	c11	1	1	0	B23C11	.
166	b37	c11	2	1	0	B37C11	.
167	b26	c12	1	1	0	B26C12	.
168	b32	c12	1	1	0	B32C12	.
169	b27	c13	1	1	0	B27C13	.
170	b41	c13	1	1	0	B41C13	.
171	b31	c14	1	1	0	B31C14	.
172	b45	c14	1	1	0	B45C14	.
173	b34	c15	2	1	0	B34C15	.
174	b40	c15	2	1	0	B40C15	.
175	b38	c17	2	1	0	B38C17	.
176	b44	c17	1	1	0	B44C17	.
177	b42	c19	1	1	0	B42C19	.
178	b48	c19	1	1	0	B48C19	.
179	b3	c2	1	1	0	B3C2	.
180	b17	c2	1	1	0	B17C2	.
181	b7	c4	1	1	0	B7C4	.
182	b21	c4	1	1	0	B21C4	.
183	b18	c8	1	1	0	B18C8	.
184	b24	c8	1	1	0	B24C8	.
185	b19	c9	1	1	0	B19C9	.
186	b33	c9	2	1	0	B33C9	.

_N_	_DEMAND_	_FLOW_	_FCOST_	_RCOST_	_STATUS_
1	10	0.5	0.5	.	NONKEY ARC BASIC
2	10	0.5	0.5	.	NONKEY ARC BASIC
3	10	0.5	0.5	.	KEY_ARC BASIC
4	10	0.5	0.5	.	KEY_ARC BASIC
5	10	0.25	0.25	.	KEY_ARC BASIC
6	10	0.5	0.5	.	KEY_ARC BASIC
7	10	0.75	0.75	.	KEY_ARC BASIC
8	10	0.25	0.25	.	NONKEY ARC BASIC
9	10	0.5	0.5	.	NONKEY ARC BASIC
10	10	0.75	0.75	.	NONKEY ARC BASIC
11	10	0.75	0.75	.	NONKEY ARC BASIC
12	10	0.75	0.75	.	NONKEY ARC BASIC



13	10	0.25	0.5	.	NONKEY ARC BASIC
14	10	0.25	0.5	.	NONKEY ARC BASIC
15	10	0.25	0.5	.	NONKEY ARC BASIC
16	10	0.75	0.75	.	KEY_ARC BASIC
17	10	0.5	0.5	.	NONKEY ARC BASIC
18	10	0.75	0.75	.	KEY_ARC BASIC
19	10	0.75	0.75	.	KEY_ARC BASIC
20	6	1	0	.	KEY_ARC BASIC
21	6	1	0	.	KEY_ARC BASIC
22	6	0.5	0	.	NONKEY ARC BASIC
23	6	0.5	0	.	NONKEY ARC BASIC
24	6	1.5	0	.	KEY_ARC BASIC
25	6	1.5	0	.	KEY_ARC BASIC
26	8	1	0	.	KEY_ARC BASIC
27	8	1.5	0	.	KEY_ARC BASIC
28	8	1.5	0	.	KEY_ARC BASIC
29	8	1.5	0	.	NONKEY ARC BASIC
30	8	0.5	0	.	NONKEY ARC BASIC
31	8	0.5	0	.	KEY_ARC BASIC
32	8	1.5	0	.	KEY_ARC BASIC
161	.	0.5	0.5	.	NONKEY ARC BASIC
162	.	0.5	0.5	.	KEY_ARC BASIC
163	.	0.75	0.75	.	KEY_ARC BASIC
164	.	0.75	0.75	.	KEY_ARC BASIC
165	.	0.25	0.25	.	KEY_ARC BASIC
166	.	0.25	0.5	.	NONKEY ARC BASIC
167	.	0.75	0.75	.	KEY_ARC BASIC
168	.	0.75	0.75	.	NONKEY ARC BASIC
169	.	0.75	0.75	.	NONKEY ARC BASIC
170	.	0.75	0.75	.	KEY_ARC BASIC
171	.	0.75	0.75	.	KEY_ARC BASIC
172	.	0.75	0.75	.	KEY_ARC BASIC
173	.	0.25	0.5	.	KEY_ARC BASIC
174	.	0.25	0.5	.	NONKEY ARC BASIC
175	.	0.25	0.5	.	NONKEY ARC BASIC
176	.	0.25	0.25	.	KEY_ARC BASIC
177	.	0.75	0.75	.	KEY_ARC BASIC
178	.	0.75	0.75	.	NONKEY ARC BASIC
179	.	0.5	0.5	.	NONKEY ARC BASIC
180	.	0.5	0.5	.	KEY_ARC BASIC
181	.	0.5	0.5	.	KEY_ARC BASIC
182	.	0.5	0.5	.	NONKEY ARC BASIC
183	.	0.75	0.75	.	KEY_ARC BASIC
184	.	0.75	0.75	.	NONKEY ARC BASIC
185	.	0.25	0.25	.	KEY_ARC BASIC
186	.	0.25	0.5	.	NONKEY ARC BASIC

*C.25 Partial Input File for Problem CELLBORDER2B, Run 7*

```
data noded;
    input _node_$ _sd_;
    cards;
S    24
DE   -10
DIR   -8
DIC   -6
;
data arcd;
input _from_$ _to_$ _cost_ _capac_ _name_$;
cards;
S m8 . 4 Sm8
S m9 . 4 Sm9
S m10 . 4 Sm10
S m11 . 4 Sm11
S m14 . 4 Sm14
S m15 . 4 Sm15
S m16 . 4 Sm16
S m17 . 4 Sm17
S m20 . 4 Sm20
S m21 . 4 Sm21
S m22 . 4 Sm22
S m23 . 4 Sm23
S m26 . 4 Sm26
S m27 . 4 Sm27
S m28 . 4 Sm28
S m29 . 4 Sm29
m8 b1 1 1 m8b1
b1 DE . 1 b1DE
m8 b2 1 1 m8b2
b2 DE . 1 b2DE
b2 c1 -1 1 b2c1
c1 DIR . 2 c1DIR
m8 b3 1 1 m8b3
b3 DE . 1 b3DE
b3 c2 -1 1 b3c2
c2 DIC . 2 c2DIC
m8 b4 1 1 m8b4
b4 DE . 1 b4DE
m9 b5 1 1 m9b5
b5 DE . 1 b5DE
m9 b6 1 1 m9b6
b6 DE . 1 b6DE
b6 c3 -1 1 b6c3
c3 DIR . 2 c3DIR
m9 b7 1 1 m9b7
b7 DE . 1 b7DE
b7 c4 -1 1 b7c4
c4 DIC . 2 c4DIC
```

m9 b8 1 1 m9b8  
 b8 DE . 1 b8DE  
 b8 c1 -1 1 b8c1  
 m10 b9 7 1 m10b9  
 b9 DE . 1 b9DE  
 m10 b10 7 1 m10b10  
 b10 DE . 1 b10DE  
 b10 c5 -1 1 b10c5  
 c5 DIR . 2 c5DIR  
 m10 b11 7 1 m10b11  
 b11 DE . 1 b11DE  
 b11 c6 -1 1 b11c6  
 c6 DIC . 2 c6DIC  
 m10 b12 7 1 m10b12  
 b12 DE . 1 b12DE  
 b12 c3 -1 1 b12c3  
 m11 b13 7 1 m11b13  
 b13 DE . 1 b13DE  
 m11 b14 7 1 m11b14  
 b14 DE . 1 b14DE  
 m11 b15 7 1 m11b15  
 b15 DE . 1 b15DE  
 b15 c7 -1 1 b15c7  
 c7 DIC . 2 c7DIC  
 m11 b16 7 1 m11b16  
 b16 DE . 1 b16DE  
 b16 c5 -1 1 b16c5  
 m14 b17 1 1 m14b17  
 b17 DE . 1 b17DE  
 b17 c2 -1 1 b17c2  
 m14 b18 1 1 m14b18  
 b18 DE . 1 b18DE  
 b18 c8 -1 1 b18c8  
 c8 DIR . 2 c8DIR  
 m14 b19 1 1 m14b19  
 b19 DE . 1 b19DE  
 b19 c9 -1 1 b19c9  
 c9 DIC . 2 c9DIC  
 m14 b20 1 1 m14b20  
 b20 DE . 1 b20DE  
 m15 b21 1 1 m15b21  
 b21 DE . 1 b21DE  
 b21 c4 -1 1 b21c4  
 m15 b22 1 1 m15b22  
 b22 DE . 1 b22DE  
 b22 c10 -1 1 b22c10  
 c10 DIR . 2 c10DIR  
 m15 b23 1 1 m15b23  
 b23 DE . 1 b23DE  
 b23 c11 -1 1 b23c11  
 c11 DIC . 2 c11DIC

m15 b24 1 1 m15b24  
 b24 DE . 1 b24DE  
 b24 c8 -1 1 b24c8  
 m16 b25 1 1 m16b25  
 b25 DE . 1 b25DE  
 b25 c6 -1 1 b25c6  
 m16 b26 1 1 m16b26  
 b26 DE . 1 b26DE  
 b26 c12 -1 1 b26c12  
 c12 DIR . 2 c12DIR  
 m16 b27 1 1 m16b27  
 b27 DE . 1 b27DE  
 b27 c13 -1 1 b27c13  
 c13 DIC . 2 c13DIC  
 m16 b28 1 1 m16b28  
 b28 DE . 1 b28DE  
 b28 c10 -1 1 b28c10  
 m17 b29 1 1 m17b29  
 b29 DE . 1 b29DE  
 b29 c7 -1 1 b29c7  
 m17 b30 1 1 m17b30  
 b30 DE . 1 b30DE  
 m17 b31 1 1 m17b31  
 b31 DE . 1 b31DE  
 b31 c14 -1 1 b31c14  
 c14 DIC . 2 c14DIC  
 m17 b32 1 1 m17b32  
 b32 DE . 1 b32DE  
 b32 c12 -1 1 b32c12  
 m20 b33 2 1 m20b33  
 b33 DE . 1 b33DE  
 b33 c9 -1 1 b33c9  
 m20 b34 2 1 m20b34  
 b34 DE . 1 b34DE  
 b34 c15 -1 1 b34c15  
 c15 DIR . 2 c15DIR  
 m20 b35 2 1 m20b35  
 b35 DE . 1 b35DE  
 b35 c16 -1 1 b35c16  
 c16 DIC . 2 c16DIC  
 m20 b36 2 1 m20b36  
 b36 DE . 1 b36DE  
 m21 b37 2 1 m21b37  
 b37 DE . 1 b37DE  
 b37 c11 -1 1 b37c11  
 m21 b38 2 1 m21b38  
 b38 DE . 1 b38DE  
 b38 c17 -1 1 b38c17  
 c17 DIR . 2 c17DIR  
 m21 b39 2 1 m21b39  
 b39 DE . 1 b39DE

b39 c18 -1 1 b39c18  
 c18 DIC . 2 c18DIC  
 m21 b40 2 1 m21b40  
 b40 DE . 1 b40DE  
 b40 c15 -1 1 b40c15  
 m22 b41 1 1 m22b41  
 b41 DE . 1 b41DE  
 b41 c13 -1 1 b41c13  
 m22 b42 1 1 m22b42  
 b42 DE . 1 b42DE  
 b42 c19 -1 1 b42c19  
 c19 DIR . 2 c19DIR  
 m22 b43 1 1 m22b43  
 b43 DE . 1 b43DE  
 b43 c20 -1 1 b43c20  
 c20 DIC . 2 c20DIC  
 m22 b44 1 1 m22b44  
 b44 DE . 1 b44DE  
 b44 c17 -1 1 b44c17  
 m23 b45 1 1 m23b45  
 b45 DE . 1 b45DE  
 b45 c14 -1 1 b45c14  
 m23 b46 1 1 m23b46  
 b46 DE . 1 b46DE  
 m23 b47 1 1 m23b47  
 b47 DE . 1 b47DE  
 b47 c21 -1 1 b47c21  
 c21 DIC . 2 c21DIC  
 m23 b48 1 1 m23b48  
 b48 DE . 1 b48DE  
 b48 c19 -1 1 b48c19  
 m26 b49 7 1 m26b49  
 b49 DE . 1 b49DE  
 b49 c16 -1 1 b49c16  
 m26 b50 7 1 m26b50  
 b50 DE . 1 b50DE  
 b50 c22 -1 1 b50c22  
 c22 DIR . 2 c22DIR  
 m26 b51 7 1 m26b51  
 b51 DE . 1 b51DE  
 m26 b52 7 1 m26b52  
 b52 DE . 1 b52DE  
 m27 b53 2 1 m27b53  
 b53 DE . 1 b53DE  
 b53 c18 -1 1 b53c18  
 m27 b54 2 1 m27b54  
 b54 DE . 1 b54DE  
 b54 c23 -1 1 b54c23  
 c23 DIR . 2 c23DIR  
 m27 b55 2 1 m27b55  
 b55 DE . 1 b55DE

m27	b56	2	1	m27b56
b56	DE	.	1	b56DE
b56	c22	-1	1	b56c22
m28	b57	3	1	m28b57
b57	DE	.	1	b57DE
b57	c20	-1	1	b57c20
m28	b58	3	1	m28b58
b58	DE	.	1	b58DE
b58	c24	-1	1	b58c24
c24	DIR	.	2	c24DIR
m28	b59	3	1	m28b59
b59	DE	.	1	b59DE
m28	b60	3	1	m28b60
b60	DE	.	1	b60DE
b60	c23	-1	1	b60c23
m29	b61	3	1	m29b61
b61	DE	.	1	b61DE
b61	c21	-1	1	b61c21
m29	b62	3	1	m29b62
b62	DE	.	1	b62DE
m29	b63	3	1	m29b63
b63	DE	.	1	b63DE
m29	b64	3	1	m29b64
b64	DE	.	1	b64DE
b64	c24	-1	1	b64c24

;

[ data cond and proc netflow not included ]

*C.26 Partial .log Output File for Problem CELLBORDER2B, Run 7*

NOTE: Number of nodes= 108 .  
NOTE: Number of supply nodes= 1 .  
NOTE: Number of demand nodes= 3 .  
NOTE: Total supply= 24 , total demand= 24 .  
NOTE: Number of arcs= 216 .  
NOTE: Number of iterations performed (neglecting any constraints)= 143 .  
NOTE: Of these, 117 were degenerate.  
NOTE: Optimum (neglecting any constraints) found.  
NOTE: Minimal total cost= 10 .  
NOTE: Number of <= side constraints= 0 .  
NOTE: Number of == side constraints= 72 .  
NOTE: Number of >= side constraints= 32 .  
NOTE: Number of arc and nonarc variable side constraint coefficients= 368 .  
NOTE: Number of iterations, optimizing with constraints= 203 .  
NOTE: Of these, 165 were degenerate.  
NOTE: Optimum reached.  
NOTE: Minimal total cost= 12 .  
NOTE: The data set WORK.SOLUTION has 216 observations and 14 variables.  
989        print arcs/nonzero; run;  
  
NOTE: The PROCEDURE NETFLOW printed pages 1-4.

*C.27 Partial Input File for Problem CELLBORDER2B, Run 8*

```
data noded;
    input _node_ $ _sd_;
    cards;
S      64
DE    -10
DIR    -8
DIC    -6
DN    -40
;
data arcd;
input _from_ $ _to_ $ _cost_ _capac_ _name_ $;
cards;
S m8 . 4 Sm8
S m9 . 4 Sm9
S m10 . 4 Sm10
S m11 . 4 Sm11
S m14 . 4 Sm14
S m15 . 4 Sm15
S m16 . 4 Sm16
S m17 . 4 Sm17
S m20 . 4 Sm20
S m21 . 4 Sm21
S m22 . 4 Sm22
S m23 . 4 Sm23
S m26 . 4 Sm26
S m27 . 4 Sm27
S m28 . 4 Sm28
S m29 . 4 Sm29
m8 b1 1 1 m8b1
b1 DE . 1 b1DE
b1 DN -2 1 b1DN
m8 b2 1 1 m8b2
b2 DE . 1 b2DE
b2 DN -2 1 b2DN
b2 c1 -1 1 b2c1
c1 DIR . 2 c1DIR
m8 b3 1 1 m8b3
b3 DE . 1 b3DE
b3 DN -2 1 b3DN
b3 c2 -1 1 b3c2
c2 DIC . 2 c2DIC
m8 b4 1 1 m8b4
b4 DE . 1 b4DE
b4 DN -2 1 b4DN
m9 b5 1 1 m9b5
b5 DE . 1 b5DE
b5 DN -2 1 b5DN
m9 b6 1 1 m9b6
b6 DE . 1 b6DE
```



b6 DN -2 1 b6DN  
b6 c3 -1 1 b6c3  
c3 DIR . 2 c3DIR  
m9 b7 i 1 m9b7  
b7 DE . 1 b7DE  
b7 DN -2 1 b7DN  
b7 c4 -1 1 b7c4  
c4 DIC . 2 c4DIC  
m9 b8 1 1 m9b8  
b8 DE . 1 b8DE  
b8 DN -2 1 b8DN  
b8 c1 -1 1 b8c1  
m10 b9 7 1 m10b9  
b9 DE . 1 b9DE  
b9 DN -14 1 b9DN  
m10 b10 7 1 m10b10  
b10 DE . 1 b10DE  
b10 DN -14 1 b10DN  
b10 c5 -1 1 b10c5  
c5 DIR . 2 c5DIR  
m10 b11 7 1 m10b11  
b11 DE . 1 b11DE  
b11 DN -14 1 b11DN  
b11 c6 -1 1 b11c6  
c6 DIC . 2 c6DIC  
m10 b12 7 1 m10b12  
b12 DE . 1 b12DE  
b12 DN -14 1 b12DN  
b12 c3 -1 1 b12c3  
m11 b13 7 1 m11b13  
b13 DE . 1 b13DE  
b13 DN -14 1 b13DN  
m11 b14 7 1 m11b14  
b14 DE . 1 b14DE  
b14 DN -14 1 b14DN  
m11 b15 7 1 m11b15  
b15 DE . 1 b15DE  
b15 DN -14 1 b15DN  
b15 c7 -1 1 b15c7  
c7 DIC . 2 c7DIC  
m11 b16 7 1 m11b16  
b16 DE . 1 b16DE  
b16 DN -14 1 b16DN  
b16 c5 -1 1 b16c5  
m14 b17 1 1 m14b17  
b17 DE . 1 b17DE  
b17 DN -2 1 b17DN  
b17 c2 -1 1 b17c2  
m14 b18 1 1 m14b18  
b18 DE . 1 b18DE  
b18 DN -2 1 b18DN

b18 c8 -1 1 b18c8  
c8 DIR . 2 c8DIR  
m14 b19 1 1 m14b19  
b19 DE . 1 b19DE  
b19 DN -2 1 b19DN  
b19 c9 -1 1 b19c9  
c9 DIC . 2 c9DIC  
m14 b20 1 1 m14b20  
b20 DE . 1 b20DE  
b20 DN -2 1 b20DN  
m15 b21 1 1 m15b21  
b21 DE . 1 b21DE  
b21 DN -2 1 b21DN  
b21 c4 -1 1 b21c4  
m15 b22 1 1 m15b22  
b22 DE . 1 b22DE  
b22 DN -2 1 b22DN  
b22 c10 -1 1 b22c10  
c10 DIR . 2 c10DIR  
m15 b23 1 1 m15b23  
b23 DE . 1 b23DE  
b23 DN -2 1 b23DN  
b23 c11 -1 1 b23c11  
c11 DIC . 2 c11DIC  
m15 b24 1 1 m15b24  
b24 DE . 1 b24DE  
b24 DN -2 1 b24DN  
b24 c8 -1 1 b24c8  
m16 b25 1 1 m16b25  
b25 DE . 1 b25DE  
b25 DN -2 1 b25DN  
b25 c6 -1 1 b25c6  
m16 b26 1 1 m16b26  
b26 DE . 1 b26DE  
b26 DN -2 1 b26DN  
b26 c12 -1 1 b26c12  
c12 DIR . 2 c12DIR  
m16 b27 1 1 m16b27  
b27 DE . 1 b27DE  
b27 DN -2 1 b27DN  
b27 c13 -1 1 b27c13  
c13 DIC . 2 c13DIC  
m16 b28 1 1 m16b28  
b28 DE . 1 b28DE  
b28 DN -2 1 b28DN  
b28 c10 -1 1 b28c10  
m17 b29 1 1 m17b29  
b29 DE . 1 b29DE  
b29 DN -2 1 b29DN  
b29 c7 -1 1 b29c7  
m17 b30 1 1 m17b30

b30 DE . 1 b30DE  
 b30 DN -2 1 b30DN  
 m17 b31 1 1 m17b31  
 b31 DE . 1 b31DE  
 b31 DN -2 1 b31DN  
 b31 c14 -1 1 b31c14  
 c14 DIC . 2 c14DIC  
 m17 b32 1 1 m17b32  
 b32 DE . 1 b32DE  
 b32 DN -2 1 b32DN  
 b32 c12 -1 1 b32c12  
 m20 b33 2 1 m20b33  
 b33 DE . 1 b33DE  
 b33 DN -4 1 b33DN  
 b33 c9 -1 1 b33c9  
 m20 b34 2 1 m20b34  
 b34 DE . 1 b34DE  
 b34 DN -4 1 b34DN  
 b34 c15 -1 1 b34c15  
 c15 DIR . 2 c15DIR  
 m20 b35 2 1 m20b35  
 b35 DE . 1 b35DE  
 b35 DN -4 1 b35DN  
 b35 c16 -1 1 b35c16  
 c16 DIC . 2 c16DIC  
 m20 b36 2 1 m20b36  
 b36 DE . 1 b36DE  
 b36 DN -4 1 b36DN  
 m21 b37 2 1 m21b37  
 b37 DE . 1 b37DE  
 b37 DN -4 1 b37DN  
 b37 c11 -1 1 b37c11  
 m21 b38 2 1 m21b38  
 b38 DE . 1 b38DE  
 b38 DN -4 1 b38DN  
 b38 c17 -1 1 b38c17  
 c17 DIR . 2 c17DIR  
 m21 b39 2 1 m21b39  
 b39 DE . 1 b39DE  
 b39 DN -4 1 b39DN  
 b39 c18 -1 1 b39c18  
 c18 DIC . 2 c18DIC  
 m21 b40 2 1 m21b40  
 b40 DE . 1 b40DE  
 b40 DN -4 1 b40DN  
 b40 c15 -1 1 b40c15  
 m22 b41 1 1 m22b41  
 b41 DE . 1 b41DE  
 b41 DN -2 1 b41DN  
 b41 c13 -1 1 b41c13  
 m22 b42 1 1 m22b42

b42 DE . 1 b42DE  
 b42 DN -2 1 b42DN  
 b42 c19 -1 1 b42c19  
 c19 DIR . 2 c19DIR  
 m22 b43 1 1 m22b43  
 b43 DE . 1 b43DE  
 b43 DN -2 1 b43DN  
 b43 c20 -1 1 b43c20  
 c20 DIC . 2 c20DIC  
 m22 b44 1 1 m22b44  
 b44 DE . 1 b44DE  
 b44 DN -2 1 b44DN  
 b44 c17 -1 1 b44c17  
 m23 b45 1 1 m23b45  
 b45 DE . 1 b45DE  
 b45 DN -2 1 b45DN  
 b45 c14 -1 1 b45c14  
 m23 b46 1 1 m23b46  
 b46 DE . 1 b46DE  
 b46 DN -2 1 b46DN  
 m23 b47 1 1 m23b47  
 b47 DE . 1 b47DE  
 b47 DN -2 1 b47DN  
 b47 c21 -1 1 b47c21  
 c21 DIC . 2 c21DIC  
 m23 b48 1 1 m23b48  
 b48 DE . 1 b48DE  
 b48 DN -2 1 b48DN  
 b48 c19 -1 1 b48c19  
 m26 b49 7 1 m26b49  
 b49 DE . 1 b49DE  
 b49 DN -14 1 b49DN  
 b49 c16 -1 1 b49c16  
 m26 b50 7 1 m26b50  
 b50 DE . 1 b50DE  
 b50 DN -14 1 b50DN  
 b50 c22 -1 1 b50c22  
 c22 DIR . 2 c22DIR  
 m26 b51 7 1 m26b51  
 b51 DE . 1 b51DE  
 b51 DN -14 1 b51DN  
 m26 b52 7 1 m26b52  
 b52 DE . 1 b52DE  
 b52 DN -14 1 b52DN  
 m27 b53 2 1 m27b53  
 b53 DE . 1 b53DE  
 b53 DN -4 1 b53DN  
 b53 c18 -1 1 b53c18  
 m27 b54 2 1 m27b54  
 b54 DE . 1 b54DE  
 b54 DN -4 1 b54DN

b54 c23 -1 1 b54c23  
 c23 DIR . 2 c23DIR  
 m27 b55 2 1 m27b55  
 b55 DE . 1 b55DE  
 b55 DN -4 1 b55DN  
 m27 b56 2 1 m27b56  
 b56 DE . 1 b56DE  
 b56 DN -4 1 b56DN  
 b56 c22 -1 1 b56c22  
 m28 b57 3 1 m28b57  
 b57 DE . 1 b57DE  
 b57 DN -6 1 b57DN  
 b57 c20 -1 1 b57c20  
 m28 b58 3 1 m28b58  
 b58 DE . 1 b58DE  
 b58 DN -6 1 b58DN  
 b58 c24 -1 1 b58c24  
 c24 DIR . 2 c24DIR  
 m28 b59 3 1 m28b59  
 b59 DE . 1 b59DE  
 b59 DN -6 1 b59DN  
 m28 b60 3 1 m28b60  
 b60 DE . 1 b60DE  
 b60 DN -6 1 b60DN  
 b60 c23 -1 1 b60c23  
 m29 b61 3 1 m29b61  
 b61 DE . 1 b61DE  
 b61 DN -6 1 b61DN  
 b61 c21 -1 1 b61c21  
 m29 b62 3 1 m29b62  
 b62 DE . 1 b62DE  
 b62 DN -6 1 b62DN  
 m29 b63 3 1 m29b63  
 b63 DE . 1 b63DE  
 b63 DN -6 1 b63DN  
 m29 b64 3 1 m29b64  
 b64 DE . 1 b64DE  
 b64 DN -6 1 b64DN  
 b64 c24 -1 1 b64c24  
 ;

[ data cond and proc netflow not included ]

*C.28 Partial .log Output File for Problem CELLBORDER2B, Run 8*

NOTE: Number of nodes= 109 .  
NOTE: Number of supply nodes= 1 .  
NOTE: Number of demand nodes= 4 .  
NOTE: Total supply= 64 , total demand= 64 .  
NOTE: Number of arcs= 280 .  
NOTE: Number of iterations performed (neglecting any constraints)= 154 .  
NOTE: Of these, 93 were degenerate.  
NOTE: Optimum (neglecting any constraints) found.  
NOTE: Minimal total cost= -130 .  
NOTE: Number of <= side constraints= 0 .  
NOTE: Number of == side constraints= 72 .  
NOTE: Number of >= side constraints= 0 .  
NOTE: Number of arc and nonarc variable side constraint coefficients= 240 .  
NOTE: Number of iterations, optimizing with constraints= 195 .  
NOTE: Of these, 177 were degenerate.  
NOTE: Optimum reached.  
NOTE: Minimal total cost= -126 .  
NOTE: The data set WORK.SOLUTION has 280 observations and 14 variables.  
981        print arcs/nonzero; run;  
  
NOTE: The PROCEDURE NETFLOW printed pages 1-8.

*C.29 Partial Input File for Problem CELLBORDER2B, Run 9*

```

data noded;
    input _node_ $ _sd_;
    cards;
S      64
DE     -10
DIR     -8
DIC     -6
DN     -40
:
data arcd;
input _from_ $ _to_ $ _cost_ _capac_ _name_ $;
cards;
S m8 . 4 Sm8
S m9 . 4 Sm9
S m10 . 4 Sm10
S m11 . 4 Sm11
S m14 . 4 Sm14
S m15 . 4 Sm15
S m16 . 4 Sm16
S m17 . 4 Sm17
S m20 . 4 Sm20
S m21 . 4 Sm21
S m22 . 4 Sm22
S m23 . 4 Sm23
S m26 . 4 Sm26
S m27 . 4 Sm27
S m28 . 4 Sm28
S m29 . 4 Sm29
b1 b1 1 1 m8b1
b1 DE . 1 b1DE
b1 DN 98 1 b1DN
m8 b2 1 1 m8b2
b2 DE . 1 b2DE
b2 DN 98 1 b2DN
b2 c1 -1 1 b2c1
c1 DIR . 2 c1DIR
m8 b3 1 1 m8b3
b3 DE . 1 b3DE
b3 DN 98 1 b3DN
b3 c2 -1 1 b3c2
c2 DIC . 2 c2DIC
m8 b4 1 1 m8b4
b4 DE . 1 b4DE
b4 DN 98 1 b4DN
m9 b5 1 1 m9b5
b5 DE . 1 b5DE
b5 DN 98 1 b5DN
m9 b6 1 1 m9b6
b6 DE . 1 b6DE

```

b6 DN 98 1 b6DN  
b6 c3 -1 1 b6c3  
c3 DIR . 2 c3DIR  
m9 b7 1 1 m9b7  
b7 DE . 1 b7DE  
b7 DN 98 1 b7DN  
b7 c4 -1 1 b7c4  
c4 DIC . 2 c4DIC  
m9 b8 1 1 m9b8  
b8 DE . 1 b8DE  
b8 DN 98 1 b8DN  
b8 c1 -1 1 b8c1  
m10 b9 7 1 m10b9  
b9 DE . 1 b9DE  
b9 DN 86 1 b9DN  
m10 b10 7 1 m10b10  
b10 DE . 1 b10DE  
b10 DN 86 1 b10DN  
b10 c5 -1 1 b10c5  
c5 DIR . 2 c5DIR  
m10 b11 7 1 m10b11  
b11 DE . 1 b11DE  
b11 DN 86 1 b11DN  
b11 c6 -1 1 b11c6  
c6 DIC . 2 c6DIC  
m10 b12 7 1 m10b12  
b12 DE . 1 b12DE  
b12 DN 86 1 b12DN  
b12 c3 -1 1 b12c3  
m11 b13 7 1 m11b13  
b13 DE . 1 b13DE  
b13 DN 86 1 b13DN  
m11 b14 7 1 m11b14  
b14 DE . 1 b14DE  
b14 DN 86 1 b14DN  
m11 b15 7 1 m11b15  
b15 DE . 1 b15DE  
b15 DN 86 1 b15DN  
b15 c7 -1 1 b15c7  
c7 DIC . 2 c7DIC  
m11 b16 7 1 m11b16  
b16 DE . 1 b16DE  
b16 DN 86 1 b16DN  
b16 c5 -1 1 b16c5  
m14 b17 1 1 m14b17  
b17 DE . 1 b17DE  
b17 DN 98 1 b17DN  
b17 c2 -1 1 b17c2  
m14 b18 1 1 m14b18  
b18 DE . 1 b18DE  
b18 DN 98 1 b18DN



b18 c8 -1 1 b18c8  
c8 DIR . 2 c8DIR  
m14 b19 1 1 m14b19  
b19 DE . 1 b19DE  
b19 DN 98 1 b19DN  
b19 c9 -1 1 b19c9  
c9 DIC . 2 c9DIC  
m14 b20 1 1 m14b20  
b20 DE . 1 b20DE  
b20 DN 98 1 b20DN  
m15 b21 1 1 m15b21  
b21 DE . 1 b21DE  
b21 DN 98 1 b21DN  
b21 c4 -1 1 b21c4  
m15 b22 1 1 m15b22  
b22 DE . 1 b22DE  
b22 DN 98 1 b22DN  
b22 c10 -1 1 b22c10  
c10 DIR . 2 c10DIR  
m15 b23 1 1 m15b23  
b23 DE . 1 b23DE  
b23 DN 98 1 b23DN  
b23 c11 -1 1 b23c11  
c11 DIC . 2 c11DIC  
m15 b24 1 1 m15b24  
b24 DE . 1 b24DE  
b24 DN 98 1 b24DN  
b24 c8 -1 1 b24c8  
m16 b25 1 1 m16b25  
b25 DE . 1 b25DE  
b25 DN 98 1 b25DN  
b25 c6 -1 1 b25c6  
m16 b26 1 1 m16b26  
b26 DE . 1 b26DE  
b26 DN 98 1 b26DN  
b26 c12 -1 1 b26c12  
c12 DIR . 2 c12DIR  
m16 b27 1 1 m16b27  
b27 DE . 1 b27DE  
b27 DN 98 1 b27DN  
b27 c13 -1 1 b27c13  
c13 DIC . 2 c13DIC  
m16 b28 1 1 m16b28  
b28 DE . 1 b28DE  
b28 DN 98 1 b28DN  
b28 c10 -1 1 b28c10  
m17 b29 1 1 m17b29  
b29 DE . 1 b29DE  
b29 DN 98 1 b29DN  
b29 c7 -1 1 b29c7  
m17 b30 1 1 m17b30

b30 DE . 1 b30DE  
 b30 DN 98 1 b30DN  
 m17 b31 1 1 m17b31  
 b31 DE . 1 b31DE  
 b31 DN 98 1 b31DN  
 b31 c14 -1 1 b31c14  
 c14 DIC . 2 c14DIC  
 m17 b32 1 1 m17b32  
 b32 DE . 1 b32DE  
 b32 DN 98 1 b32DN  
 b32 c12 -1 1 b32c12  
 m20 b33 2 1 m20b33  
 b33 DE . 1 b33DE  
 b33 DN 96 1 b33DN  
 b33 c9 -1 1 b33c9  
 m20 b34 2 1 m20b34  
 b34 DE . 1 b34DE  
 b34 DN 96 1 b34DN  
 b34 c15 -1 1 b34c15  
 c15 DIR . 2 c15DIR  
 m20 b35 2 1 m20b35  
 b35 DE . 1 b35DE  
 b35 DN 96 1 b35DN  
 b35 c16 -1 1 b35c16  
 c16 DIC . 2 c16DIC  
 m20 b36 2 1 m20b36  
 b36 DE . 1 b36DE  
 b36 DN 96 1 b36DN  
 m21 b37 2 1 m21b37  
 b37 DE . 1 b37DE  
 b37 DN 96 1 b37DN  
 b37 c11 -1 1 b37c11  
 m21 b38 2 1 m21b38  
 b38 DE . 1 b38DE  
 b38 DN 96 1 b38DN  
 b38 c17 -1 1 b38c17  
 c17 DIR . 2 c17DIR  
 m21 b39 2 1 m21b39  
 b39 DE . 1 b39DE  
 b39 DN 96 1 b39DN  
 b39 c18 -1 1 b39c18  
 c18 DIC . 2 c18DIC  
 m21 b40 2 1 m21b40  
 b40 DE . 1 b40DE  
 b40 DN 96 1 b40DN  
 b40 c15 -1 1 b40c15  
 m22 b41 1 1 m22b41  
 b41 DE . 1 b41DE  
 b41 DN 98 1 b41DN  
 b41 c13 -1 1 b41c13  
 m22 b42 1 1 m22b42

b42 DE . 1 b42DE  
 b42 DN 98 1 b42DN  
 b42 c19 -1 1 b42c19  
 c19 DIR . 2 c19DIR  
 m22 b43 1 1 m22b43  
 b43 DE . 1 b43DE  
 b43 DN 98 1 b43DN  
 b43 c20 -1 1 b43c20  
 c20 DIC . 2 c20DIC  
 m22 b44 1 1 m22b44  
 b44 DE . 1 b44DE  
 b44 DN 98 1 b44DN  
 b44 c17 -1 1 b44c17  
 m23 b45 1 1 m23b45  
 b45 DE . 1 b45DE  
 b45 DN 98 1 b45DN  
 b45 c14 -1 1 b45c14  
 m23 b46 1 1 m23b46  
 b46 DE . 1 b46DE  
 b46 DN 98 1 b46DN  
 m23 b47 1 1 m23b47  
 b47 DE . 1 b47DE  
 b47 DN 98 1 b47DN  
 b47 c21 -1 1 b47c21  
 c21 DIC . 2 c21DIC  
 m23 b48 1 1 m23b48  
 b48 DE . 1 b48DE  
 b48 DN 98 1 b48DN  
 b48 c19 -1 1 b48c19  
 m26 b49 7 1 m26b49  
 b49 DE . 1 b49DE  
 b49 DN 86 1 b49DN  
 b49 c16 -1 1 b49c16  
 m26 b50 7 1 m26b50  
 b50 DE . 1 b50DE  
 b50 DN 86 1 b50DN  
 b50 c22 -1 1 b50c22  
 c22 DIR . 2 c22DIR  
 m26 b51 7 1 m26b51  
 b51 DE . 1 b51DE  
 b51 DN 86 1 b51DN  
 m26 b52 7 1 m26b52  
 b52 DE . 1 b52DE  
 b52 DN 86 1 b52DN  
 m27 b53 2 1 m27b53  
 b53 DE . 1 b53DE  
 b53 DN 96 1 b53DN  
 b53 c18 -1 1 b53c18  
 m27 b54 2 1 m27b54  
 b54 DE . 1 b54DE  
 b54 DN 96 1 b54DN

b54 c23 -1 1 b54c23  
 c23 DIR . 2 c23DIR  
 m27 b55 2 1 m27b55  
 b55 DE . 1 b55DE  
 b55 DN 96 1 b55DN  
 m27 b56 2 1 m27b56  
 b56 DE . 1 b56DE  
 b56 DN 96 1 b56DN  
 b56 c22 -1 1 b56c22  
 m28 b57 3 1 m28b57  
 b57 DE . 1 b57DE  
 b57 DN 94 1 b57DN  
 b57 c20 -1 1 b57c20  
 m28 b58 3 1 m28b58  
 b58 DE . 1 b58DE  
 b58 DN 94 1 b58DN  
 b58 c24 -1 1 b58c24  
 c24 DIR . 2 c24DIR  
 m28 b59 3 1 m28b59  
 b59 DE . 1 b59DE  
 b59 DN 94 1 b59DN  
 m28 b60 3 1 m28b60  
 b60 DE . 1 b60DE  
 b60 DN 94 1 b60DN  
 b60 c23 -1 1 b60c23  
 m29 b61 3 1 m29b61  
 b61 DE . 1 b61DE  
 b61 DN 94 1 b61DN  
 b61 c21 -1 1 b61c21  
 m29 b62 3 1 m29b62  
 b62 DE . 1 b62DE  
 b62 DN 94 1 b62DN  
 m29 b63 3 1 m29b63  
 b63 DE . 1 b63DE  
 b63 DN 94 1 b63DN  
 m29 b64 3 1 m29b64  
 b64 DE . 1 b64DE  
 b64 DN 94 1 b64DN  
 b64 c24 -1 1 b64c24  
 ;

[ data cond and proc netflow not included ]

*C.30 Partial .log Output File for Problem CELLBORDER2B, Run 9*

NOTE: Number of nodes= 109 .  
NOTE: Number of supply nodes= 1 .  
NOTE: Number of demand nodes= 4 .  
NOTE: Total supply= 64 , total demand= 64 .  
NOTE: Number of arcs= 280 .  
NOTE: Number of iterations performed (neglecting any constraints)= 186 .  
NOTE: Of these, 125 were degenerate.  
NOTE: Optimum (neglecting any constraints) found.  
NOTE: Minimal total cost= 3870 .  
NOTE: Number of <= side constraints= 0 .  
NOTE: Number of == side constraints= 72 .  
NOTE: Number of >= side constraints= 0 .  
NOTE: Number of arc and nonarc variable side constraint coefficients= 240 .  
NOTE: Number of iterations, optimizing with constraints= 213 .  
NOTE: Of these, 196 were degenerate.  
NOTE: Optimum reached.  
NOTE: Minimal total cost= 3874 .  
NOTE: The data set WORK.SOLUTION has 280 observations and 14 variables.  
982        print arcs/nonzero; run;  
  
983  
NOTE: The PROCEDURE NETFLOW printed pages 1-8.

*C.31 Partial Input File for Problem CELLBORDER2B, Run 10*

```
data noded;
    input _node_$ _sd_;
    cards;
S      24
DE     -10
DI     -14
;
data arcd;
input _from_$ _to_$ _cost_ _capac_ _name_$;
cards;
S m8 . 4 Sm8
S m9 . 4 Sm9
S m10 . 4 Sm10
S m11 . 4 Sm11
S m14 . 4 Sm14
S m15 . 4 Sm15
S m16 . 4 Sm16
S m17 . 4 Sm17
S m20 . 4 Sm20
S m21 . 4 Sm21
S m22 . 4 Sm22
S m23 . 4 Sm23
S m26 . 4 Sm26
S m27 . 4 Sm27
S m28 . 4 Sm28
S m29 . 4 Sm29
m8 b1 . 1 m8b1
b1 DE . 1 b1DE
m8 b2 . 1 m8b2
b2 DE 1 1 b2DE
b2 c1 1 1 b2c1
c1 DI . 2 c1DI
m8 b3 . 1 m8b3
b3 DE 1 1 b3DE
b3 c2 1 1 b3c2
c2 DI . 2 c2DI
m8 b4 . 1 m8b4
b4 DE 1 1 b4DE
m9 b5 . 1 m9b5
b5 DE 1 1 b5DE
m9 b6 . 1 m9b6
b6 DE 1 1 b6DE
b6 c3 1 1 b6c3
c3 DI . 2 c3DI
m9 b7 . 1 m9b7
b7 DE 1 1 b7DE
b7 c4 1 1 b7c4
c4 DI . 2 c4DI
m9 b8 . 1 m9b8
```

b8 DE 1 1 b8DE  
 b8 c1 1 1 b8c1  
 m10 b9 . 1 m10b9  
 b9 DE 7 1 b9DE  
 m10 b10 . 1 m10b10  
 b10 DE 7 1 b10DE  
 b10 c5 7 1 b10c5  
 c5 DI . 2 c5DI  
 m10 b11 . 1 m10b11  
 b11 DE 7 1 b11DE  
 b11 c6 7 1 b11c6  
 c6 DI . 2 c6DI  
 m10 b12 . 1 m10b12  
 b12 DE 7 1 b12DE  
 b12 c3 7 1 b12c3  
 m11 b13 . 1 m11b13  
 b13 DE 7 1 b13DE  
 m11 b14 . 1 m11b14  
 b14 DE 7 1 b14DE  
 m11 b15 . 1 m11b15  
 b15 DE 7 1 b15DE  
 b15 c7 7 1 b15c7  
 c7 DI . 2 c7DI  
 m11 b16 . 1 m11b16  
 b16 DE 7 1 b16DE  
 b16 c5 7 1 b16c5  
 m14 b17 . 1 m14b17  
 b17 DE 1 1 b17DE  
 b17 c2 1 1 b17c2  
 m14 b18 . 1 m14b18  
 b18 DE 1 1 b18DE  
 b18 c8 1 1 b18c8  
 c8 DI . 2 c8DI  
 m14 b19 . 1 m14b19  
 b19 DE 1 1 b19DE  
 b19 c9 1 1 b19c9  
 c9 DI . 2 c9DI  
 m14 b20 . 1 m14b20  
 b20 DE 1 1 b20DE  
 m15 b21 . 1 m15b21  
 b21 DE 1 1 b21DE  
 b21 c4 1 1 b21c4  
 m15 b22 . 1 m15b22  
 b22 DE 1 1 b22DE  
 b22 c10 1 1 b22c10  
 c10 DI . 2 c10DI  
 m15 b23 . 1 m15b23  
 b23 DE 1 1 b23DE  
 b23 c11 1 1 b23c11  
 c11 DI . 2 c11DI  
 m15 b24 . 1 m15b24

b24	DE	1	1	b24DE
b24	c8	1	1	b24c8
m16	b25	.	1	m16b25
b25	DE	1	1	b25DE
b25	c6	1	1	b25c6
m16	b26	.	1	m16b26
b26	DE	1	1	b26DE
b26	c12	1	1	b26c12
c12	DI	.	2	c12DI
m16	b27	.	1	m16b27
b27	DE	1	1	b27DE
b27	c13	1	1	b27c13
c13	DI	.	2	c13DI
m16	b28	.	1	m16b28
b28	DE	1	1	b28DE
b28	c10	1	1	b28c10
m17	b29	.	1	m17b29
b29	DE	1	1	b29DE
b29	c7	1	1	b29c7
m17	b30	.	1	m17b30
b30	DE	1	1	b30DE
m17	b31	.	1	m17b31
b31	DE	1	1	b31DE
b31	c14	1	1	b31c14
c14	DI	.	2	c14DI
m17	b32	.	1	m17b32
b32	DE	1	1	b32DE
b32	c12	1	1	b32c12
m20	b33	.	1	m20b33
b33	DE	2	1	b33DE
b33	c9	2	1	b33c9
m20	b34	.	1	m20b34
b34	DE	2	1	b34DE
b34	c15	2	1	b34c15
c15	DI	.	2	c15DI
m20	b35	.	1	m20b35
b35	DE	2	1	b35DE
b35	c16	2	1	b35c16
c16	DI	.	2	c16DI
m20	b36	.	1	m20b36
b36	DE	2	1	b36DE
m21	b37	.	1	m21b37
b37	DE	2	1	b37DE
b37	c11	2	1	b37c11
m21	b38	.	1	m21b38
b38	DE	2	1	b38DE
b38	c17	2	1	b38c17
c17	DI	.	2	c17DI
m21	b39	.	1	m21b39
b39	DE	2	1	b39DE
b39	c18	2	1	b39c18



c18 DI . 2 c18DI  
m21 b40 . 1 m21b40  
b40 DE 2 1 b40DE  
b40 c15 2 1 b40c15  
m22 b41 . 1 m22b41  
b41 DE 1 1 b41DE  
b41 c13 1 1 b41c13  
m22 b42 . 1 m22b42  
b42 DE 1 1 b42DE  
b42 c19 1 1 b42c19  
c19 DI . 2 c19DI  
m22 b43 . 1 m22b43  
b43 DE 1 1 b43DE  
b43 c20 1 1 b43c20  
c20 DI . 2 c20DI  
m22 b44 . 1 m22b44  
b44 DE 1 1 b44DE  
b44 c17 1 1 b44c17  
m23 b45 . 1 m23b45  
b45 DE 1 1 b45DE  
b45 c14 1 1 b45c14  
m23 b46 . 1 m23b46  
b46 DE 1 1 b46DE  
m23 b47 . 1 m23b47  
b47 DE 1 1 b47DE  
b47 c21 1 1 b47c21  
c21 DI . 2 c21DI  
m23 b48 . 1 m23b48  
b48 DE 1 1 b48DE  
b48 c19 1 1 b48c19  
m26 b49 . 1 m26b49  
b49 DE 7 1 b49DE  
b49 c16 7 1 b49c16  
m26 b50 . 1 m26b50  
b50 DE 7 1 b50DE  
b50 c22 7 1 b50c22  
c22 DI . 2 c22DI  
m26 b51 . 1 m26b51  
b51 DE 7 1 b51DE  
m26 b52 . 1 m26b52  
b52 DE 7 1 b52DE  
m27 b53 . 1 m27b53  
b53 DE 2 1 b53DE  
b53 c18 2 1 b53c18  
m27 b54 . 1 m27b54  
b54 DE 2 1 b54DE  
b54 c23 2 1 b54c23  
c23 DI . 2 c23DI  
m27 b55 . 1 m27b55  
b55 DE 2 1 b55DE  
m27 b56 . 1 m27b56

b56 DE 2 1 b56DE  
 b56 c22 2 1 b56c22  
 m28 b57 . 1 m28b57  
 b57 DE 3 1 b57DE  
 b57 c20 3 1 b57c20  
 m28 b58 . 1 m28b58  
 b58 DE 3 1 b58DE  
 b58 c24 3 1 b58c24  
 c24 DI . 2 c24DI  
 m28 b59 . 1 m28b59  
 b59 DE 3 1 b59DE  
 m28 b60 . 1 m28b60  
 b60 DE 3 1 b60DE  
 b60 c23 3 1 b60c23  
 m29 b61 . 1 m29b61  
 b61 DE 3 1 b61DE  
 b61 c21 3 1 b61c21  
 m29 b62 . 1 m29b62  
 b62 DE 3 1 b62DE  
 m29 b63 . 1 m29b63  
 b63 DE 3 1 b63DE  
 m29 b64 . 1 m29b64  
 b64 DE 3 1 b64DE  
 b64 c24 3 1 b64c24  
 ;

[ data cond and netflow not included ]

### C.32 Partial .log and .lis Output Files for Problem CELLBORDER2B, Run 10

NOTE: Number of nodes= 107 .  
 NOTE: Number of supply nodes= 1 .  
 NOTE: Number of demand nodes= 2 .  
 NOTE: Total supply= 24 , total demand= 24 .  
 NOTE: Number of arcs= 216 .  
 NOTE: Number of iterations performed (neglecting any constraints)= 131 .  
 NOTE: Of these, 109 were degenerate.  
 NOTE: Optimum (neglecting any constraints) found.  
 NOTE: Minimal total cost= 24 .  
 NOTE: Number of <= side constraints= 0 .  
 NOTE: Number of == side constraints= 72 .  
 NOTE: Number of >= side constraints= 32 .  
 NOTE: Number of arc and nonarc variable side constraint coefficients= 368 .  
 NOTE: Number of iterations, optimizing with constraints= 141 .  
 NOTE: Of these, 119 were degenerate.  
 NOTE: Optimum reached.  
 NOTE: Minimal total cost= 25.142857143 .  
 NOTE: The data set WORK.SOLUTION has 216 observations and 14 variables.  
 \*8C print arcs/nonzero; run;

NOTE: The PROCEDURE NETFLOW printed pages 1-4.

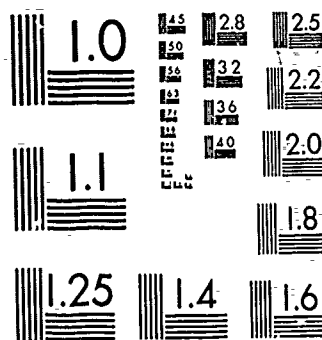
_N_	_FROM_	_TO_	_COST_	_CAPAC_	_LO_	_NAME_	_SUPPLY_
97	S	m14	0	4	0	SM14	24
98	S	m15	0	4	0	SM15	24
99	S	m16	0	4	0	SM16	24
100	S	m17	0	4	0	SM17	24
101	S	m20	0	4	0	SM20	24
102	S	m21	0	4	0	SM21	24
103	S	m22	0	4	0	SM22	24
104	S	m23	0	4	0	SM23	24
105	S	m8	0	4	0	SM8	24
106	S	m9	0	4	0	SM9	24

_N_	_DEMAND_	_FLOW_	_FCOST_	_RCOST_	_STATUS_
97	.	2.8571428571	0	.	KEY_ARC BASIC
98	.	2.8571428571	0	.	KEY_ARC BASIC
99	.	2.8571428571	0	.	KEY_ARC BASIC
100	.	2.8571428571	0	.	KEY_ARC BASIC
101	.	0.5714285714	0	.	KEY_ARC BASIC
102	.	0.5714285714	0	.	KEY_ARC BASIC
103	.	2.8571428571	0	.	KEY_ARC BASIC
104	.	2.8571428571	0	.	KEY_ARC BASIC
105	.	2.8571428571	0	.	KEY_ARC BASIC
106	.	2.8571428571	0	.	KEY_ARC BASIC

## Appendix D. *Pixel Subregion Allocation Computer Programs, Input and Output Files*

### D.1 *The "IMAGING" System Program*

```
$ ! -----
$ ! FILE:IMAGING.COM
$ ! DESCRIPTION: Executes the GAMS input file generator (a Pascal
$ !   program named SUBNETS.EXE), then GAMS itself, and finally
$ !   the solution program (pascal program SOLUTION.EXE).
$ ! This file is executed interactively using the command:
$ !
$ ! @imaging
$ !
$ ! ENVIRONMENT: VAX/VMS (hercules)
$ ! AUTHOR: Thomas G. Reed
$ ! DATE: OCT 91
$ ! -----
$ CLEAR
$ DEFINE/USER_MODE SYS$INPUT SYS$COMMAND
$ ! run the executable file to query the user and set up the GAMS file
$ RUN SUBNETS
$ ! get access to the GAMS library
$ GETGAMS
$ ! run GAMS
$ GAMS NSA.DAT
$ ! rename the GAMS output
$ REN NSA.LIS LISTING.DAT
$ RUN SOLUTION
$ ! get rid of old data files (too many in the directory cause problems)
$ DELETE WORKING.DAT;*
$ DELETE NSA.DAT;*
$ DELETE XVAR.DAT;*
$ PURGE LISTING.DAT
$ EXIT
```



MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

## D.2 The "SOLUTION" Pascal Program

```
program solution (input, output, rcdata, listing, xvar, compactim);

{ Program to read from the GAMS output file LISTING.DAT
  (which is a copy of the NSA.LIS generated by GAMS) and
  present the output in a 2-D format on screen and also
  in a data file "compactim" (which can be re-displayed at a later time
  for visual comparison purposes with the original image). }

var
  rcdata, listing, xvar, compactim: text;
  rows, columns: integer;

{***** PROCEDURE FIND_DATA *****)
procedure find_data;

const
  key1 = '----';
  key2 = ' VAR';
  key3 = ' X';

type
  string = packed array [1..4] of char;

var
  gotit: boolean;
  i, safety: integer;
  section: array [1..3] of string;      {used to find the X var output section}

begin {find_data}
  gotit := false;
  safety := 1;
  while ( (not (gotit)) and (safety<1000) )do    {go to VAR X portion of output file}
  begin
    readln (listing);
    for i := 1 to 3 do
    begin
      read (listing, section[i])
    end; {for}
    safety := safety+1;
    if (section[1]=key1) and (section[2]=key2) and (section[3]=key3) then gotit := true
  end {while}                                {we are at the X variable output section}
end; {find_data}

{***** PROCEDURE XVAR_FILE *****)
procedure xvar_file(var rows, columns: integer);

type
```

```

string = packed array [1..8] of char;

var
  gotit: boolean;
  i, j: integer;
  part: array [1..3] of string;
  ch: array [1..10] of char;

begin {xvar_file}
  read (rcdata, rows);           {determine number of rows and columns}
  read (rcdata, columns);
  i := 1;
  while ( (not eof (listing)) and (i<=((columns+2)*(rows+2))) ) do
    begin
      read (listing, ch[1]);
      read (listing, ch[2]);
      while ( (ch[1]>='1')and(ch[1]<='9') or (ch[2]>='1')and(ch[2]<='9') ) do
        {check for data line}
        for j := 1 to 3 do read (listing, part[j]);
        writeln (xvar, part[3]);   {write X var data to the PARTIAL file}
        i := i+1;
        readln (listing);
        read (listing, ch[1]);
        read (listing, ch[2])
      end; {while ch}
      readln (listing)
    end {while not eof}
  end; {xvar_file}

{***** PROCEDURE ANSWER *****)}

procedure answer(var rows, columns: integer);

var
  pixel: array [1..30, 1..30] of integer;
  i, j, k, lastcol, area: integer;
  gotit: boolean;
  ch: array [1..10] of char;

begin {answer}

  for i := 1 to (rows+2) do
    begin
      for j := 1 to (columns+2) do
        begin
          pixel[i,j] := 88
          {set all pixels to 88 initially}
        end {for j}
      end; {for i}

    lastcol := 3;
    if (rows+2) > 9 then lastcol := lastcol + 1;

```

```

if (columns+2) > 9 then lastcol := lastcol + 1;
for i := 1 to (rows+2) do
begin
  for j := 1 to (columns+2) do
  begin
    gotit := false;
    while (not (gotit)) do
    begin
      k := 1;
      while ( (k <= lastcol) and not (eof(xvar)) ) do
      begin
        read (xvar, ch[k]);
        if (ch[k] = '.') then gotit := true;  {if there is a 1 anywhere on}
        if (ch[k] = '1') then pixel[i,j] := 1; {the output line, we have}
        k := k+1;                               {found a subregion pixel}
      end; {while}
      readln (xvar)
    end {while}
  end {for j}
end; {for i}

writeln;
writeln;
writeln ('*****');
writeln ('SOLUTION IMAGE:');           {show user what image looks like}
writeln;
write ('    1');                       {label for column 1}
for j := 2 to columns do              {labels for columns 2 and up}
begin
  write (j:3);
end;
writeln;
writeln;
area := 0;
for i := 2 to rows+1 do               {write output in matrix form}
begin
  write ((i-1):2);                     {labels for row numbers}
  for j := 2 to columns+1 do
  begin
    if pixel[i,j] = 1 then area := area+1; {determine area}
    write (pixel[i,j]:3);               {solution pixels written to screen}
    write (compactim, pixel[i,j]:3)     {solution pixels written to file}
  end;
  writeln;
  writeln (compactim)
end;
writeln;
writeln ('NOTE: Pixels denoted with a 1 represent actual subregion pixels');
writeln ('as dictated by your spectral and spatial requirements; those');
writeln ('denoted with 88 are non-subregion pixels. ');
writeln;

```



```

        writeln ('The subregion pixels comprise a total area of: ', area:5);
        writeln ('*****');
        writeln
end; {answer}

{***** MAIN PROGRAM : SOLUTION *****}

begin {solution}
    reset (listing);
    find_data;                      {run FIND_DATA proc to search thru GAMS output file}
    reset (rcdata);
    rewrite (xvar);
    xvar_file (rows, columns);      {run XVAR_FILE proc to copy X values over}
    reset (xvar);
    rewrite (compactim);
    answer (rows, columns);         {run ANSWER proc to output to screen the solution}
end. {solution}

```

### *D.3 The Complete Screen Output of the Pseudo-Image Low Compactness Example Run*

\$ Qimaging

\*\*\*\*\*

THE USE OF NETWORKS FOR SUBREGION ALLOCATION:  
BASED ON CONCEPTS DEVELOPED BY BENABDALLAH AND WRIGHT  
AND IMPLEMENTATION PROCEDURES DEVELOPED BY BURKE.  
CREATED BY THOMAS G. REED

\*\*\*\*\*

This program will use data contained in the text file  
"IMAGE.DAT". The SAS portion of the program will create  
for later execution the SAS file "NSA" (which stands for  
"Network Subregion Allocation").

#### PURPOSE

The purpose of this program is twofold:

- (1) determine which pixels of an image make up the  
subregion(s) of interest as specified by the spectral  
and spatial requirements of the program user;

OR

- (2) determine the minimum cost single subregion of a data matrix  
representing cost, population, grey values, distance, etc.;

#### INPUT

The data matrix MUST be contained in a TEXT file called IMAGE.DAT.  
The user will be queried for additional information as needed.

#### OUTPUT

The output of this program depends on the options chosen by the  
user. Working files used for data manipulation and either a GAMS  
or SAS input file are common outputs for all options.

\*\*\*\*\*

NOTE: the image cannot exceed 20 cells by 20 cells, thank you.

Enter the number of ROWS of the image/data

12

Enter the number of COLUMNS of the image/data

12

\*\*\*\*\*  
ORIGINAL DATA MATRIX:

11	21	12	13	22	14	15	23	16	17	18	19
11	12	24	13	14	25	15	16	17	26	18	19
11	12	13	14	15	16	17	27	28	29	18	19
11	12	13	14	21	22	23	24	25	26	27	15
16	17	18	28	19	29	21	22	23	24	25	11
12	13	14	25	26	27	28	29	21	22	23	24
15	16	17	18	25	26	27	28	19	11	12	13
14	15	16	17	18	22	19	11	23	24	12	13
14	15	16	25	17	18	19	11	26	12	13	14
15	16	17	18	19	11	12	13	14	15	16	17
18	19	11	22	12	13	14	23	15	16	17	18
19	11	12	13	14	15	16	17	18	19	11	12

\*\*\*\*\*

# of rows is: 12; columns is: 12; size is: 144

\*\*\*\*\*

Enter one of the following program options:

1: IMAGE SUBREGION DETERMINATION  
(based on spatial and spectral requirements)

2: MIN-COST SUBREGION DETERMINATION

1

\*\*\*\*\*

NOTE: The subregion minimum grey value you are about to enter must not exceed the subregion maximum grey value. Also, both must be between 0 and 255, inclusive.

Enter the MINIMUM grey value of the subregion of interest  
20

Enter the MAXIMUM grey value of the subregion of interest  
30

\*\*\*\*\*  
 WORKING IMAGE WITH "FRAME":

```

88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 1 88 88 1 88 88 1 88 88 88 88
88 88 88 1 88 88 1 88 88 88 1 88 88 88
88 88 88 88 88 88 88 88 1 1 1 88 88 88
88 88 88 88 88 1 1 1 1 1 1 1 88 88
88 88 88 88 1 88 1 1 1 1 1 1 1 88 88
88 88 88 88 1 1 1 1 1 1 1 1 1 88
88 88 88 88 88 1 1 1 1 88 88 88 88 88
88 88 88 88 88 88 1 88 88 1 1 88 88 88
88 88 88 88 1 88 88 88 88 1 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 1 88 88 88 1 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88

```

NOTE: Pixels denoted with a 1 represent subregion candidate pixels based on your spectral requirements.

The candidate pixels comprise a total area of: 43

\*\*\*\*\*

Enter one of the following compactness/contiguity options:

\*\*\*\*\*

- 1: None: no compactness/contiguity requirements  
 (show all pixels within the grey range of interest)
- 2: Marginal compactness/contiguity requirements  
 (delete single pixel renegades:)  
 (pixels must have at least one 1st- or 2nd-order neighbor)
- 3: Low compactness/contiguity requirements  
 (pixels must have at least one 1st-order neighbor)
- 4: Medium compactness/contiguity requirements  
 (pixels must have at least two total neighbors)
- 5: High compactness/contiguity requirements  
 (pixels must have at least two 1st-order neighbors)

3.

\*\*\*\*\* FINDING OPTIMUM SOLUTION . . . PLEASE WAIT \*\*\*\*\*

--- GAMS 2.20

--- Compiling then executing GS091D:[TREED.CFILES]NSA.DAT;94

--- Output is being written to GS091D:[TREED.CFILES]NSA.LIS;30

--- Output line length will be 79 characters

--- Starting ZOOM

--- Resuming execution

--- All done

\*\*\*\*\*

SOLUTION IMAGE:

	1	2	3	4	5	6	7	8	9	10	11	12
1	88	88	88	88	88	88	88	88	88	88	88	88
2	88	88	88	88	88	88	88	88	88	1	88	88
3	88	88	88	88	88	88	88	1	1	1	88	88
4	88	88	88	88	1	1	1	1	1	1	1	88
5	88	88	88	1	88	1	1	1	1	1	1	88
6	88	88	88	1	1	1	1	1	1	1	1	1
7	88	88	88	88	1	1	1	1	88	88	88	88
8	88	88	88	88	88	1	88	88	1	1	88	88
9	88	88	88	88	88	88	88	88	1	88	88	88
10	88	88	88	88	88	88	88	88	88	88	88	88
11	88	88	88	88	88	88	88	88	88	88	88	88
12	88	88	88	88	88	88	88	88	88	88	88	88

NOTE: Pixels denoted with a 1 represent actual subregion pixels as dictated by your spectral and spatial requirements; those denoted with 88 are non-subregion pixels.

The subregion pixels comprise a total area of: 35

\*\*\*\*\*

# *D.4 The GAMS Input File for the Low Compactness Example Run*

SETS

I image rows with frame / 1 \* 14/  
J image cols with frame / 1 \* 14/;

TABLE C(I,J) cost matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	88	88	88	88	88	88	88	88	88	88	88	88	88	88
2	88	88	1	88	88	1	88	88	1	88	88	88	88	88
3	88	88	88	1	88	88	1	88	88	88	1	88	88	88
4	88	88	88	88	88	88	88	88	1	1	1	88	88	88
5	88	88	88	88	88	1	1	1	1	1	1	1	88	88
6	88	88	88	88	1	88	1	1	1	1	1	1	88	88
7	88	88	88	88	1	1	1	1	1	1	1	1	1	88
8	88	88	88	88	88	1	1	1	1	88	88	88	88	88
9	88	88	88	88	88	88	1	88	88	1	1	88	88	88
10	88	88	88	88	1	88	88	88	88	1	88	88	88	88
11	88	88	88	88	88	88	88	88	88	88	88	88	88	88
12	88	88	88	88	1	88	88	88	1	88	88	88	88	88
13	88	88	88	88	88	88	88	88	88	88	88	88	88	88
14	88	88	88	88	88	88	88	88	88	88	88	88	88	88

VARIABLES

Z total area of allocated cells

X(I,J) the image cell;

BINARY VARIABLE X;

EQUATIONS

SIZE OBJECTIVE FUNCTION

C1(I,J) CONSTRAINT FOR NONSUBREGION ALLOCATION

C2(I,J) CONSTRAINT FOR CONTIGUITY;

SIZE..

Z =E= SUM((I,J), X(I,J));

C1(I,J)\$(C(I,J) NE 1)..

X(I,J) =E= 0;

C2(I,J)\$(C(I,J) EQ 1)..

X(I-1,J) + X(I,J+1) + X(I+1,J) + X(I,J-1) - X(I,J) =G= 0;

MODEL SUBREGION /ALL/;

OPTIONS ITERLIM = 100000, RESLIM = 1000000, WORK = 50000;

OPTIONS LIMROW = 0, LIMCOL = 0;

SOLVE SUBREGION USING MIP MAXIMIZING Z;

OPTION MIP=ZOOM;

DISPLAY X.L;

# D:5 The Partial Screen Output of the Pseudo-Image High Compactness Example Run

\*\*\*\*\*

Enter one of the following compactness/contiguity options:

\*\*\*\*\*

- 1: None: no compactness/contigui requirements  
(show all pixels within the grey range of interest)
- 2: Marginal compactness/contiguity requirements  
(delete single pixel renegades:)  
(pixels must have at least one 1st- or 2nd-order neighbor)
- 3: Low compactness/contiguity requirements  
(pixels must have at least one 1st-order neighbor)
- 4: Medium compactness/contiguity requirements  
(pixels must have at least two total neighbors)
- 5: High compactness/contiguity requirements  
(pixels must have at least two 1st-order neighbors)

5

\*\*\*\*\* FINDING OPTIMUM SOLUTION . . . PLEASE WAIT \*\*\*\*\*

\*\*\*\*\*

SOLUTION IMAGE:

	1	2	3	4	5	6	7	8	9	10	11	12
1	88	88	88	88	88	88	88	88	88	88	88	88
2	88	88	88	88	88	88	88	88	88	88	88	88
3	88	88	88	88	88	88	88	1	1	1	88	88
4	88	88	88	88	88	1	1	1	1	1	1	88
5	88	88	88	88	88	1	1	1	1	1	1	88
6	88	88	88	88	1	1	1	1	1	1	1	88
7	88	88	88	88	1	1	1	1	88	88	88	88
8	88	88	88	88	88	88	88	88	88	88	88	88
9	88	88	88	88	88	88	88	88	88	88	88	88
10	88	88	88	88	88	88	88	88	88	88	88	88
11	88	88	88	88	88	88	88	88	88	88	88	88
12	88	88	88	88	88	88	88	88	88	88	88	88

NOTE: Pixels denoted with a 1 represent actual subregion pixels as dictated by your spectral and spatial requirements; those denoted with 88 are non-subregion pixels.

The subregion pixels comprise a total area of: 26

\*\*\*\*\*

## D.6 The Partial Screen Output of the Real Image High Compactness Example Run

\*\*\*\*\*

NOTE: The subregion minimum grey value you are about to enter must not exceed the subregion maximum grey value.

Also, both must be between 0 and 255, inclusive.

Enter the MINIMUM grey value of the subregion of interest

200

Enter the MAXIMUM grey value of the subregion of interest

255

\*\*\*\*\*

WORKING IMAGE WITH "FRAME":

```
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 1 1 1 1 1 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 1 88 1 1 1 1 1 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 1 1 1 1 1 1 1 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 1 88 1 1 1 1 1 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 1 88
88 1 1 88 88 1 1 88 88 88 88 88 88 88 1 1 1 88 88 1 88
88 1 1 1 88 1 1 88 88 88 88 88 88 88 1 1 1 1 88 88 88 88
88 1 1 1 1 1 1 1 88 88 88 88 88 88 1 1 1 1 88 88 88 88
88 1 1 1 1 1 1 1 88 88 88 88 88 88 1 1 88 88 88 88 88
88 1 1 1 1 1 1 1 1 88 88 88 88 88 88 1 1 1 88 88 88 88
88 1 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 1 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 1 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 1 1 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
```

NOTE: Pixels denoted with a 1 represent subregion candidate pixels based on your spectral requirements.

The candidate pixels comprise a total area of: 94

\*\*\*\*\*

Enter one of the following compactness/contiguity options:

\*\*\*\*\*

- 1: None: no compactness/contiguity requirements  
(show all pixels within the grey range of interest)
- 2: Marginal compactness/contiguity requirements  
(delete single pixel renegades:)  
(pixels must have at least one 1st- or 2nd-order neighbor)



- 3: Low compactness/contiguity requirements  
(pixels must have at least one 1st-order neighbor)
- 4: Medium compactness/contiguity requirements  
(pixels must have at least two total neighbors)
- 5: High compactness/contiguity requirements  
(pixels must have at least two 1st-order neighbors)

5

\*\*\*\*\* FINDING OPTIMUM SOLUTION . . . PLEASE WAIT \*\*\*\*\*

```

--- GAMS 2.20
--- Compiling then executing GSO91D:[TREED.CFILES]NSA.DAT;4
--- Output is being written to GSO91D:[TREED.CFILES]NSA.LIS;31
--- Output line length will be 79 characters
--- Starting ZOOM
--- Resuming execution
--- All done

```

\*\*\*\*\*  
SOLUTION IMAGE:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
2	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	1	1	1	1	1
3	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	1	1	1	1	1
4	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	1	1	1	1	1
5	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	1	1	1	1	1
6	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
7	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
8	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
9	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
10	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
11	1	1	88	88	1	1	88	88	88	88	88	88	88	88	1	1	1	88	88	88
12	1	1	1	88	1	1	88	88	88	88	88	88	88	1	1	1	1	88	88	88
13	1	1	1	1	1	1	1	88	88	88	88	88	88	1	1	1	1	88	88	88
14	1	1	1	1	1	1	1	88	88	88	88	88	88	88	1	1	88	88	88	88
15	1	1	1	1	1	1	1	88	88	88	88	88	88	88	1	1	88	88	88	88
16	1	1	1	1	1	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
17	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
18	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
19	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88
20	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88

NOTE: Pixels denoted with a 1 represent actual subregion pixels as dictated by your spectral and spatial requirements; those denoted with 88 are non-subregion pixels.

The subregion pixels comprise a total area of: 70

\*\*\*\*\*

### *D.7 The Partial Output File of the Real Image Example Run*

COMPILATION TIME = 0.550 SECONDS

#### MODEL STATISTICS

BLOCKS OF EQUATIONS	3	SINGLE EQUATIONS	485
BLOCKS OF VARIABLES	2	SINGLE VARIABLES	485
NON ZERO ELEMENTS	1345	DISCRETE VARIABLES	484

GENERATION TIME = 2.030 SECONDS

EXECUTION TIME = 2.590 SECONDS

#### S O L V E S U M M A R Y

MODEL	SUBREGION	OBJECTIVE	Z
TYPE	MIP	DIRECTION	MAXIMIZE
SOLVER	ZOOM	FROM LINE	57

\*\*\*\* SOLVER STATUS 1 NORMAL COMPLETION  
\*\*\*\* MODEL STATUS 8 INTEGER SOLUTION  
\*\*\*\* OBJECTIVE VALUE 70.0000

RESOURCE USAGE, LIMIT	1904.940	1000000.000
ITERATION COUNT, LIMIT	35380	1000000

Z O C M / X M P --- Version 2.1 Oct 1988

Courtesy of Dr Roy E. Marsten,  
Department of Management Information Systems,  
University of Arizona,  
Tucson Arizona 85721, U.S.A.

No options file found - using defaults.

Work space needed (estimate) -- 148185 words.  
Work space available -- 148185 words.

Note: The stopping tolerance is satisfied,  
but the solution may not be optimal.

No solution better than 77.500000 can exist.  
(absolute distance 7.5000000 , OPTCA 0.00000000E+00 )  
(relative distance 9.6774E-02, OPTCR 0.1000 )

The branch and bound tree contained 156 nodes (max. 50000 nodes).

Iterations: Initial LP	528,	Time: 9.09998
Heuristic	33,	410.430
Branch and bound	34818,	1483.26
Final LP	1,	0.290039

## Appendix E. *The Burke Implementation of the B&W Models*

### *E.1 A Few Words Concerning the Work of Burke*

Most of the specific details of Burke's work (4) applicable to this thesis have already been explained in various chapters of this thesis. To note is the fact that Burke also worked with the implementation of the B&W districting model and heuristics which were not examined in this thesis research. His specific contributions which marked the departure point of this thesis effort were that he:

- Determined an appropriate solver package to use;
- Determined a way to properly handle perimeter cells to ensure proper border calculations for the model constraints and objectives;
- Developed computer programs to implement the B&W models;
- Experimented with many problems to explore the efficient frontier of the example solutions, the utility of the B&W models, and to make adjustments and corrections to the constraints that made up the model.

## E.2 The "MSA" System Program

```

$ ! -----
$ ! FILE:          MSA.COM
$ ! DESCRIPTION:   Executes the GAMS input file generator then GAMS itself.
$ !               This file is executed interactively using the command:
$ !
$ !               @msa
$ ! ENVIRONMENT:   VAX/VMS (hercules)
$ ! AUTHOR:        DAN BURKE
$ ! DATE:          7/15/91
$ ! -----
$ CLEAR
$
$ ! display the following title on CRT
$ TYPE SYS$INPUT
*****
*               MULTIPLE SUBREGION ALLOCATION MODELS               *
*                                                                 *
*               BY BENABDALLAH AND WRIGHT                         *
*                                                                 *
*               IMPLEMENTED BY DAN BURKE                          *
*****
$
$ ! prompt for filenames
$ INQUIRE FILENAME "Enter Image File (Cost Matrix)"
$ INQUIRE GAMSFILE "Enter GAMS File"
$
$ ! define keyboard as data entry source
$ DEFINE/USER_MODE SYS$INPUT SYS$COMMAND
$
$ ! define symbol for GAMS input file generator
$ INPUT == "$ STAFF:[DBURKE]MSA_INPUT.EXE"
$
$ ! execute GAMS input file generator
$ INPUT 'FILENAME' 'GAMSFILE'
$
$ ! execute GAMS
$ GAMS 'GAMSFILE'
$
$ ! delete GAMS input file (since GAMS output file contains a copy)
$ DELETE 'GAMSFILE';*
$
$ EXIT

```

## Bibliography

1. Benabdallah, Salah and Jeff R. Wright. "Multiple Subregion Allocation Models." Unpublished manuscript, 1990.
2. Bénié, Goze Bertin and others. "A Comparison of Four Segmentation Algorithms in the Context of Agricultural Remote Sensing," *ISPRS Journal of Photogrammetry and Remote Sensing*, 44:1-13 (September 1989).
3. Bruynooghe, M. "Recent Results in Clustering Very Large Data Sets." *Astronomy from Large Databases: ESO Conference and Workshop Proceedings*, 28. 101-106. Germany: European Southern Observatory, January 1988.
4. Burke, Dan, Research Assistant. Personal interviews, instruction, and handouts, July 1991. Air Force Institute of Technology, Wright-Patterson AFB, OH.
5. Celenk, Mehmet and Prabhashankar Lakshman. "Parallel Implementation of the Split and Merge Algorithm on Hypercube Processors for Object Detection and Recognition." *Proceedings of the SPIE: Applications of Artificial Intelligence VII*, 1095. 251-262. Washington: SPIE, March 1989.
6. Chan, Yupo. "Remote Sensing and Geographic Information Systems: Applications to Spatial-Temporal Analysis." Unpublished manuscript, 1991.
7. Chan, Yupo. *OPER 7.67: Networks and Combinatorial Optimization*. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, Summer, 1991. Class lectures and handouts.
8. Chan, Yupo and Thomas S. Kelso. *OPER 6.37: Analysis and Modelling of Spatial-Temporal Information*. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, Summer, 1991. Class lectures and handouts.
9. Haddon, John F. and James F. Boyce. "The Unification of Region and Edge Information for Image Segmentation." *Proceedings of the SPIE: Multispectral Image Processing and Enhancement*, 933. 15-22. Washington: SPIE, April 1988.
10. Joshi, Ashok Kumar. "Automatic Detection of Lineaments from Landsat Data." *Proceedings of the IGARSS '89 12th Canadian Symposium on Remote Sensing*. 85-88. Canada: IGARSS, July 1989.
11. Klasse, Dorothy Fay. *Use of Satellite Imagery to Monitor the Oasis Agriculture in the Turpan Depression, Xinjiang Uygur Autonomous Region, People's Republic of China-A Case Study*. MS thesis, University of Hawaii, May 1990. DTIC# AD-A220 613.

12. Lalitha, L. "A Technique for Road Detection from High Resolution Satellite Images." *Proceedings of the IGARSS '89 12th Canadian Symposium on Remote Sensing*. 2246-2249. Canada: IGARSS, July 1989.
13. Lee, Changkyu and others. "Image Segmentation Using the Morphological Pyramid." *Proceedings of the SPIE: Applications of Artificial Intelligence VII, 1095*. 208-221. Washington: SPIE, March 1989.
14. Meyer, Robert H. and Kevin K. Tong. "Detection of Airplanes in Aerial Imagery." *Proceedings of the SPIE: Applications of Digital Image Processing XII, 1153*. 456-467. Washington: SPIE, August 1989.
15. Samy, R.A. and A. Lucas. "I.R. Aerial Images Segmentation Based on Correlation of Local Histograms." *Proceedings of the SPIE: Applications of Digital Image Processing XII, 1153*. 477-484. Washington: SPIE, August 1989.
16. Smith, Stanley H. and Youcef Oubraham. "Automatic Object Detection and Recognition Using Simple Geometric Primitives." *Proceedings of the SPIE: Applications of Digital Image Processing XII, 1153*. 363-375. Washington: SPIE, August 1989.
17. Sudibjo, E.R. and others. "Digital Enhancement of STAR-1 SAR Imagery for Linear Feature Extraction." *Proceedings of the IGARSS '89 12th Canadian Symposium on Remote Sensing*. 2242-2245. Canada: IGARSS, July 1989.
18. Wang, Jinfei and Philip J. Howarth. "Edge Following as Graph Searching and Hough Transform Algorithms for Lineament Detection." *Proceedings of the IGARSS '89 12th Canadian Symposium on Remote Sensing*. 93-96. Canada: IGARSS, July 1989.
19. Zelek, John S. "Computer-Aided Linear Planimetric Feature Extraction." *Proceedings of the IGARSS '89 12th Canadian Symposium on Remote Sensing*. 2250-2253. Canada: IGARSS, July 1989.
20. Zhu, Zhipu and Yongmin Kim. "Algorithm for Automatic Road Recognition on Digitized Map Images," *Optical Engineering*, 28(9):949-954 (September 1989).

# REPORT DOCUMENTATION PAGE

Form Approved

OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 4500 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE December 1991		3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE BINARY PROGRAMMING MODELS OF SPATIAL PATTERN RECOGNITION: APPLICATIONS IN REMOTE SENSING IMAGE ANALYSIS				5. FUNDING NUMBERS	
6. AUTHOR(S) Thomas G. Reed, Captain, USAF					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, WPAFB OH 45433-6583				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GSO/ENS/91D-15	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES					
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited				12b. DISTRIBUTION CODE	
<p>13. ABSTRACT (Maximum 200 words)</p> <p>The major purpose of this investigation was to implement subregion allocation objectives using a network model based on an existing subregion allocation binary programming model (Benabdallah and Wright (B&amp;W), 1990), the ultimate goal being the application of subregion allocation concepts towards the spatial analysis of satellite imagery.</p> <p>The multi-objective aspects of subregion allocation can be accomplished via a network formulation, a formulation vastly simpler in complexity than the binary programming models previously used. Without a network programming package that could maintain integral flows, however, deriving the solution was a tiresome task for the user. Nonetheless, several new concepts and advantages to using networks were discussed and demonstrated to the degree possible.</p> <p>Spatial analysis could benefit from applying a modified version of the B&amp;W model to the pixels of satellite imagery in a way which takes advantage of the inherent contiguity of natural terrain subregions. Trending and forecasting of subregion changes, tasks that rely on acquiring data in a consistent manner image after image, could benefit due to the fact that major spatial characteristics of subregions could be extracted, and minor spatial changes could be removed.</p>					
14. SUBJECT TERMS Geodesy, Pattern Recognition, Linear Programming, Network Flows				15. NUMBER OF PAGES 304	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL		